Predicate and propositional logic

Logic helps us communicate precisely: program specifications, database queries, circuits

Proposition: a statement that is either true of false.
Example: 2+3=5 (true) 1+1=3 (false)
For every non-negative integer n, n2+n+41 is prime (false)
Goldbach's conjecture: Every even integer greater than 2 is the sum of 2 prime numbers (unknown true or false)
Ambiguous sentences ()

Connectives can change or combine propositions.
Boolean variable: a variable that has only two possible values, true and false. (denoted by 1 and 0)
Truth Table: columns for different combinations of Boolean variables, one row for each possible assignment of values to the Boolean variables.

Negation: NOT(P), ~P, ¬P, P with a bar on top of it
P and NOT(NOT(P)) are logically equivalent. (have identical columns in the truth table)

Conjunction: P AND Q, P∧Q, P&Q, P·Q
Disjunction: P OR Q, P∨Q, P+Q
Exclusive-OR: P XOR Q, P⊕Q

P	NOT(P)	Q	P AND Q	P OR Q	P XOR Q	P IMPLIES Q	P IFF Q	NOT(P) OR Q
F	T	F	F	F	T	T(vacuously true)	T (!!)	T
F		T	F	T	T	T(vacuously true)	F	T
T	F	F	F	T	F	F	F	F
T		T	T	T	T	T	T	T

De Morgan's Laws:
NOT(P OR Q) and NOT(P) AND NOT(Q) are logically equivalent.
NOT(P AND Q) and NOT(P) OR NOT(Q) are logically equivalent.

Implication: P IMPLIES Q, P→Q, P⇒Q, P⊃Q (symbol to denote superset)
Hypothesis/Antecedent P, Conclusion/Consequent Q
P IMPLIES Q is logically equivalent to NOT(P) OR Q (implies is not associative but OR is)

P implies Q = P requires Q
If P, (then) Q = Q if P = P only if Q.
When[ever] P, [then] Q = Q when[ever] P.
P only when Q = Only when Q, P.
P is sufficient/enough for Q = For Q, P is sufficient/enough.
Q is necessary for P = For P, Q is necessary. (cannot P without Q)

Contrapositive: NOT(Q) IMPLIES NOT(P)
Converse: Q IMPLIES P converse of converse of implication is the original implication
contrapositive of contrapositive of implication is NOT(NOT(P)) IMPLIES NOT(NOT(Q)) = original implication

Equivalence:
P IFF Q, P↔Q, P⇔Q, P=Q, P≡Q, P is equivalent to Q, P if and only if Q, P is necessary and sufficient for Q

Predicate: a proposition whose truth depends on the value of one or more variables; a function whose range is {T, F}

employee	gender	salary
Andy	M	0
Donna	F	3000
Leslie	F	5000
Ron	M	5700
Tom	M	1800

Let a(e)='employee e made at least 1000', let E denote set of employees.
Then a: E→{T, F}
a(e)=T if e=Ron; a(e)=F is e=Andy

Let s(e)='salary of employee e' (Not a predicate) 's(e)≥1000' is a predicate

Universal Quantification ∀x∈D, p(x) "For all"

Every employee made at least 1000.
Each employee made at least 1000.
All employees made at least 1000.
Employees made at least 1000.

Proposition ∀e∈E, a(e) is false (Andy made less than 1000)

To show that ∀x∈D, p(x) is false, give a counterexample: an x∈D such that p(x) is false.
∀x∈D, p(x) is true when p(x) is true for every x∈D.

Existential Quantification ∃x∈D, p(x) "there exists"

There is an employee who made at least 1000.
There exists an employee who made at least 1000.
Some employee made at least 1000.
For some employee e, e made at least 1000.
At least one employee made at least 1000.

To show that ∃x∈D, p(x) is true, give an example: an x∈D such that p(x) is true.
Provided D≠∅, if ∀x∈D, p(x) is true, then ∃x∈D, p(x) is true.
∃x∈D, p(x) is false when p(x) is false for every x∈D.

NOT(∃x∈D, p(x)) is logically equivalent to ∀x∈D, NOT(p(x))
NOT(∃x∈D, p(x)) IFF ∀x∈D, NOT(p(x)) is true

∀x∈D, p(x) is false, when there exists a counterexample.
NOT(∀x∈D, p(x)) IFF ∃x∈D, NOT(p(x)) is true

An employee made at least 1000. ✘ ∀e∈E, a(e) or ∃e∈E, a(e) (ambiguous)

If (∀x∈D. p(x)) OR (∀x∈D. q(x)) is true, then ∀x∈D, (p(x) OR q(x)) is true; The converse is not always true.
(∀x∈D. p(x)) AND (∀x∈D. q(x)) is equivalent to ∀x∈D. (p(x) AND q(x)).
If ∃x∈D, (p(x) AND q(x)) is true, then (∃x∈D. p(x)) AND (∃x∈D. q(x)) is true. The converse is not always true.
(∃x∈D. p(x)) AND (∃y∈D. q(y)) is equivalent to ∃x∈D. ∃y∈D. (p(x) AND q(y)).
(∀x∈D. p(x)) OR (∀y∈D. q(y)) is equivalent to ∀x∈D. ∀y∈D. (p(x) OR q(y)).
∃x∈D, (p(x) OR q(x)) is equivalent to (∃x∈D. p(x)) OR (∃x∈D. q(x)).

Some x with property p also has property q. ∃x∈D, (p(x) AND q(x))
Every x with property p also has property q. ∀x∈D, (p(x) IMPLIES q(x))
There is no x with property p that also has property q. NOT(∃x∈D, (p(x) AND q(x)))
Every x with property p does not have property q. ∀x∈D, (p(x) IMPLIES NOT q(x))
Not every x with property p also has property q. NOT(∀x∈D, (p(x) IMPLIES q(x)))
There is some x with property p that does not have property q. ∃x∈D, (p(x) AND NOT(q(x)))

Mixing Quantifiers:
Every output has some input that connects to it. ∀o∈O, ∃i∈I, connect(i, o)
Some input connects to all outputs. ∃i∈I, ∀o∈O, connect(i, o)
[∃x∈D1, ∀y∈D2, p(x, y)] IMPLIES [∀y∈D2, ∃x∈D1, p(x, y)]

Example: let M: R×C→{T, F}

Every row of M contains T. $\forall x \in R, \exists y \in C, M(x, y)$

$NOT(\forall x \in R, \exists y \in C, M(x, y))$ Not every row of M contains T. (= Some rows of M are all F.)

$\exists x \in R, NOT(\exists y \in C, M(x, y))$ Some row of M does not contain T. (= There is one row all F.)

$\exists x \in R, \forall y \in C, NOT(M(x, y))$ Some row of M is all F.

$NOT(\forall x \in R, \exists y \in C, M(x, y))$ is equivalent to $\exists x \in R, \forall y \in C, NOT(M(x, y))$

The negation of "Some column of M is entire T" is "Every column of M contains F".

Propositional Formula: an expression build up from Boolean variables using connectives (AND, OR, NOT, IMPLIES, IFF, XOR). It does not contain predicates or quantifiers.

A propositional formula is **valid** or a **tautology** if all its truth table entries are true.

P OR NOT(P) NOT(P OR Q) IFF (NOT(P) AND NOT(Q))

A IFF B, where A and B are logically equivalent

A propositional formula is **unsatisfiable** or a **contradiction** if all its truth table entries are false.

P AND NOT(P)NOT(A), where A is a tautology

A propositional formula is **satisfiable** if it is not unsatisfiable.

A truth assignment is a function from a set of propositional variables to {T, F}.

Example τ: {P, Q}→{T, F}

Each row in a truth table corresponds to a different truth assignment with the same domain.

An algorithm to determine if a propositional formula is satisfiable: construct its truth table

If a formula has n variables, there are 2n rows in its truth table.

Satisfiability Problem(SAT): decide whether a given propositional formula is satisfiable.

Input: a propositional formula

Output: Yes if the formula is satisfiable; No if it is unsatisfiable.

P = all decision problems that can be solved in polynomial time

NP = all decision problems that can be verified in polynomial time

SAT∈NP

Example: (P OR NOT(Q)) AND (NOT(P) OR Q) is satisfiable.

Because τ((P OR NOT(Q)) AND (NOT(P) OR Q)) = T where τ(P) = F and τ(Q) = F.

Theorem: SAT∈P if and only if P=NP.

Literal: a variable or the negation of a variable

Note: it is easy to solve satisfiability for a formula that is the conjunction of literals.

Example: P AND NOT(Q) AND R is satisfiable

P AND NOT(Q) AND R AND NOT(P) is unsatisfiable

Check that the formula does not contain a variable and its negation.

A propositional formula is in **disjunctive normal form** if it is a disjunction of conjunctions of literals.

Theorem: every proportional formula is logically equivalent to a propositional formula in DNF.

A propositional formula is in **conjunctive normal form** if it is a conjunction of disjunctions of literals.

A **clause** is a disjunction of literals. A CNF formula is a conjunction of clauses.

Theorem: every propositional formula is logically equivalent to a propositional formula in CNF.

~~ construct DNF for the complement, negate it, apply deMorgan's Law and simplify (replace all occurrences of NOT(NOT(X)) with X)

CNF-SAT: decide whether a given propositional formula in CNF is satisfiable.

Input: a propositional formula in CNF

Output: YES if the formula is satisfiable, NO if it is unsatisfiable.

CNF-SAT is just as hard as SAT.

Predicate Logic Formula: an expression build up from predicate symbols, each of which has a fixed number of arguments, using connectives and universal and existential quantifiers.

The arguments of predicate symbols are variables and constants from specific domains.

$\forall x \in D, [P(x, y, 0)$ IMPLIES $\exists y \in D, (S(x, y)$ AND $G(y))]$, where $0 \in D$, $G: D \to \{T, F\}$ $S: D \times D \to \{T, F\}$ $P: D \times D \times D \to \{T, F\}$

A constant symbol denotes one particular element in a domain.

A variable can be used to denote any element in a domain.

An **occurrence** of a variable x is **quantified** if it occurs within a sub-formula of the form ∀x∈D. E or ∃x∈D. E;
Otherwise, it is **unquantified/free**. Constants are never quantified.

∀x ∈ D.[P(x,x,z) IMPLIES ∃y ∈ D.(S(x,y) AND G(y))]
all occurrences of x and y are quantified, the occurrence of z is free.
∀x ∈ D.[P(x,y,z) IMPLIES ∃y ∈ D.(S(x,y) AND G(y))] the first occurrence of y is free
Better: ∀x ∈ D.[P(x,y,z) IMPLIES ∃w ∈ D.(S(x,w) AND G(w))]
DON'T use the same symbol for a free variable and a quantified variable in the same formula.
∀y ∈ D.[P(x,y,z) IMPLIES ∃y ∈ D.(S(x,y) AND G(y))] occurrences of x and z are free
the occurrences of y are quantified
Better: ∀y ∈ D.[P(x,y,z) IMPLIES ∃w ∈ D.(S(x,w) AND G(w))]
DON'T use the same symbol for nested quantified variables.
The truth of a predicate logic formula depends on: what the predicate symbols mean, what the domains are and what the constant symbols refer to.
p: D→{T, F} ∀x∈D, p(x) True if D={1,3,5} and p(x)='x is odd' False if D={1,2,3} and p(x)='x is odd'

An **Interpretation** of a predicate logic formula with no free variables consists of:
a nonempty set for each domain in the formula,
a function from the relevant domain to the relevant range for each function symbol in the formula, and
a domain element for each constant symbol. (element in relevant domain)
In particular, each predicate symbol in the formula must have a function from the relevant domain to {T, F}.

If D = ϕ, then ∀x∈D.p(x) is always vacuously true, ∃x∈D.p(x) is always false.

An interpretation of a predicate logic formula consists of:
a nonempty set for each domain in the formula,
a function from the relevant domain to the relevant range for each function symbol in the formula,
a domain element for each constant symbol, and
an element of the relevant domain for each free variable.

A **valuation** maps each free variable to a domain element.

A predicate logic formula is valid/a tautology if it true for all interpretations.
A predicate logic formula is satisfiable if it is true for some interpretation.
A predicate logic formula is unsatisfiable if it is false for all interpretations.

E logically implies E′ means that E′ is true in every interpretation that makes E true.
E and E′ are logically equivalent if E logically implies E′ and E′ logically implies E.

∃x∈D.(p(x) OR q(x)) is logically equivalent to (∃x∈D.p(x)) OR (∃x∈D.q(x)).
∃x∈D.(p(x) AND q(x)) logically implies that (∃x∈D.p(x)) AND (∃x∈D.q(x))
If x does not occur in E, then ∃x∈D.(E AND q(x)) is logically equivalent to E AND (∃x∈D.q(x)).

∀x∈D.(p(x) AND q(x)) is logically equivalent to (∀x∈D.p(x)) AND (∀x∈D.q(x))
(∀x∈D.p(x)) OR (∀x∈D.q(x)) logically implies that ∀x∈D.(p(x) OR q(x))
If x does not occur in E, then ∀x∈D.(E OR q(x)) is logically equivalent to E OR (∀x∈D.q(x)).

If x does not occur in E, then
∃x∈D.(E IMPLIES q(x)) is logically equivalent to E IMPLIES ∃x∈D.q(x).
∀x∈D.(E IMPLIES q(x)) is logically equivalent to E IMPLIES ∀x∈D.q(x).
∃x∈D.(p(x) IMPLIES E) is logically equivalent to (∀x∈D.p(x)) IMPLIES E
∀x∈D.(p(x) IMPLIES E) is logically equivalent to (∃x∈D.p(x)) IMPLIES E.

A predicate logic formula is in **prenex normal form** if and only if it is of the form

$Q_1x_1\in D_1.\ Q_2x_2\in D_2.\ \cdots\ Q_kx_k\in D_k.\ E(x_1,\ldots,x_k)$,
where $E(x_1,...,x_k)$ is a formula without quantifiers and for all i=1,…,k, Q_i is either ∀ or ∃ (quantifiers).
Any predicate logic formula can be converted to prenex normal form by applying a sequence of transformations.

NOT(∃x∈D.p(x)) can be transformed to ∀x∈D.NOT(p(x)).
NOT(∀x∈D.p(x)) can be transformed to ∃x∈D.NOT(p(x)).

(∃x∈D.p(x)) OR (∃x∈D.q(x)) can be transformed to ∃x∈D.(p(x) OR q(x))
(∀x∈D.p(x)) AND (∀x∈D.q(x)) can be transformed to ∀x∈D.(p(x) AND q(x))
(∃x∈D.p(x)) AND (∃x∈D.q(x)) CANNOT be transformed to ∃x∈D.(p(x) AND q(x))
(∀x∈D.p(x)) OR (∀x∈D.q(x)) CANNOT be transformed to ∀x∈D.(p(x) OR q(x))

If x does not occur in E, then
E AND ∃x∈D.q(x) can be transformed to ∃x∈D.(E AND q(x))
(∃x∈D.p(x)) AND E can be transformed to ∃x∈D.(p(x) AND E)
E AND ∀x∈D.q(x) can be transformed to ∀x∈D.(E AND q(x))
(∀x∈D.p(x)) AND E can be transformed to ∀x∈D.(p(x) AND E)

If x does not occur in E, then
E OR ∃x∈D.q(x) can be transformed to ∃x∈D.(E OR q(x))
(∃x∈D.p(x)) OR E can be transformed to ∃x∈D.(p(x) OR E)
E OR ∀x∈D.q(x) can be transformed to ∀x∈D.(E OR q(x))
(∀x∈D.p(x)) OR E can be transformed to ∀x∈D.(p(x) OR E)

If x does not occur in E, then
E IMPLIES ∃x∈D.q(x) can be transformed to ∃x∈D.(E IMPLIES q(x))
(∃x∈D.p(x)) IMPLIES E can be transformed to ∀x∈D.(p(x) IMPLIES E)
E IMPLIES ∀x∈D.q(x) can be transformed to ∀x∈D.(E IMPLIES q(x))
(∀x∈D.p(x)) IMPLIES E can be transformed to ∃x∈D.(p(x) IMPLIES E)

Proof

Proposition: a statement that is either true or false.
Axiom: a proposition that we agree is true.
Proof: a convincing argument that a proposition is true.
It consists of a sequence of axioms, previously proved propositions, and logical deductions.
A **logical deduction** uses an **inference rule** to prove a new proposition from axioms and previously proved propositions.

Substitution:

Let R be a tautology that contains propositional variable P.
Let R' is the formula obtained by replacing every occurrence of P in R by the formula (Q),
then R' is a tautology.
R = (A OR P) IMPLIES (P OR A)
Q = C AND D
R' = (A OR **(C AND D)**) IMPLIES ((**C AND D**) OR A)
(A OR (C AND D)) IMPLIES (P OR A) is not a tautology
Q = ∀e∈E. a(e)
R' = (A OR (∀**e∈E. a(e)**)) IMPLIES ((∀**e∈E. a(e)**) OR A)

Let S' be a formula that is logically equivalent to S.
If S is a sub-formula of R and R' is a formula obtained by replacing some occurrences of S in R by S',
then R' is logically equivalent to R.
S = NOT(A) AND NOT(B)

S' = NOT(A OR B)
R = (NOT(A) AND NOT(B)) XOR (B IFF (NOT(A) AND NOT(B)))
R' = **NOT(A OR B)** XOR (B IFF (NOT(A) AND NOT(B))) = (NOT(A) AND NOT(B)) XOR (B IFF (**NOT(A OR B)**))
R' = **NOT(A OR B)** XOR (B IFF (**NOT(A OR B)**)) R is logically equivalent to R' (all of three)

Modus Ponens: If P and P IMPLIES Q are axioms or previously proved propositions, then Q is true.

Example:
1.∀e∈E. l(e) axiom
2.(∀e∈E. l(e)) IMPLIES ∃e∈E. l(e) tautology
3.∃e∈E. l(e) modus ponens 1,2

Formal Proof:
—Number each line
—Write one proposition per line
—Justify each line

If P and P IMPLIES Q are true propositions, then Q is a true proposition.
Not an example:
If he is a criminal, he has something to hide. He has something to hide.
Therefore he is a criminal.
From the axioms P IMPLIES Q and Q, you can't conclude P.

Transitivity: If P IMPLIES Q and Q IMPLIES R are axioms or previously proved propositions, then P IMPLIES R is true.

Example:
If you study hard, you'll learn the material.
If you learn the material, you'll pass the course.
Therefore, if you study hard, you'll pass the course.

Direct Proof of Implication:

Assume P
⋮
Q
Therefore P IMPLIES Q
Example: proof of transitivity
1.Assume P
2.P IMPLIES Q axiom
3.Q modus ponens 1,2
4.Q IMPLIES R axiom
5.R modus ponens 3,4
6.Therefore P IMPLIES R direct proof 1,5

Indirect Proof of Implication:

P IMPLIES Q is logically equivalent to NOT(Q) IMPLIES NOT(P),
so proving NOT(Q) IMPLIES NOT(P) proves P IMPLIES Q.
Assume NOT(Q)
⋮
NOT(P)
Hence NOT(Q) IMPLIES NOT(P).
Therefore P IMPLIES Q.

Lemma: If x is even, then x2 is even.
Proof:
Assume x is even
Then x=2k for some integer k
so x2=(2k)2=2×(2k2), which is even

Lemma: If x2 is even, then x is even. (direct proof is hard, lets try indirect proof)
Proof:
Suppose x is not even.
Then x is odd,
so x=2k+1 for some integer k.
Hence x2=(2k+1)2=4k2+4k+1=2(2k2+2k)+1, which is odd.
Therefore x2 is not even.

Proof of a Disjunction:

⋮
P
Therefore P OR Q.

Proof of a Conjunction:

⋮
P
⋮
Q
Therefore P AND Q.

Use of Conjunction:

P AND Q
(P AND Q) IMPLIES P tautology
P

Proof by Contradiction: Q AND NOT(Q) is a contradiction, so NOT(Q AND NOT(Q)) is a tautology.

Assume NOT(P)
⋮
Q
⋮
NOT(Q)
Q AND NOT(Q) proof of conjunction
NOT(Q AND NOT(Q)) IMPLIES P indirect proof
NOT(Q AND NOT (Q)) tautology
Therefore P modus ponens

Theorem: $\sqrt{2}$ is irrational
Proof: To obtain a contradiction, assume $\sqrt{2}$ is rational.
Then there exist relatively prime positive integers x and y, such that $\sqrt{2}=x/y$.
Since $x^2=(\sqrt{2}y)^2=2y^2$, x^2 is even.
From the lemma, x is even.
Thus, there exist $k \in Z+$ such that $x=2k$,
so $2y^2=x^2=(2k)^2=4k^2$ and $y^2=2k^2$
Therefore y^2 is even.
From the lemma, y is even.
Since 2 divides both x and y, they are not relatively prime.
This is a contradiction. Hence $\sqrt{2}$ is irrational.

Proof of Equivalences: P IFF Q is logically equivalent to (P IMPLIES Q) AND (Q IMPLIES P)

Prove P IMPLIES Q and Q IMPLIES P separately.
To prove P IFF Q, Q IFF R, and P IFF R, it suffices to prove P IMPLIES Q, Q IMPLIES R and R IMPLIES P.
Note: Proving P IMPLIES Q, Q IMPLIES R, and P IMPLIES R is not sufficient.

Use of Equivalence:

P IFF Q
P IMPLIES Q

Proof by Cases:

⋮
P_1 IMPLIES Q
⋮
P_n IMPLIES Q
Therefore (P_1 OR… OR P_n) IMPLIES Q
Note: the cases P_1, … P_n can overlap.

P_1 OR … OR Pn
⋮
P_1 IMPLIES Q
⋮
P_n IMPLIES Q
Therefore (P_1 OR… OR P_n) IMPLIES Q
Hence Q.

P IMPLIES (P_1 OR … OR P_n)
⋮
P_1 IMPLIES Q
⋮
P_n IMPLIES Q
Therefore (P_1 OR… OR P_n) IMPLIES Q
Hence P IMPLIES Q.

LEMMA: P IMPLIES ((P AND C) OR (P AND NOT(C)))
1.Assume P
2.C OR NOT C tautology
    3.Assume C
    4.P AND C proof of conjunction 1,3
    5. (P AND C) OR (P AND NOT(C)) proof of disjunction 4
6.C IMPLIES ((P AND C) OR (P AND NOT(C))) direct proof 3,5
    7.Assume NOT(C)
    8.P AND NOT(C) proof of conjunction 1,7
    9.(P AND C) OR (P AND NOT(C)) proof of disjunction 8
10. NOT(C) IMPLIES ((P AND C) OR (P AND NOT(C))) direct proof 7,9
11.(P AND C) OR (P AND NOT (C)) proof by cases 2,6,10
12.P IMPLIES ((P AND C) OR (P AND NOT (C))) direct proof 1,11

Lemma: $\lfloor (n + 1)/2 \rfloor = \lceil n/2 \rceil$.
Proof:
Suppose n is even. Then n=2k for some integer k, so $\lfloor (n + 1)/2 \rfloor = \lfloor (2k + 1)/2 \rfloor = \lfloor k+1/2 \rfloor = k = \lceil k \rceil = \lceil 2k/2 \rceil = \lceil n/2 \rceil$
Then n is even IMPLIES $\lfloor (n + 1)/2 \rfloor = \lceil n/2 \rceil$.
Suppose n is odd. Then n=2k+1 for some integer k, so $\lfloor (n + 1)/2 \rfloor = \lfloor (2k + 2)/2 \rfloor = \lfloor k + 1 \rfloor = k+1 = \lceil k+1/2 \rceil = \lceil (2k+1)/2 \rceil = \lceil n/2 \rceil$
Then n is odd IMPLIES $\lfloor (n + 1)/2 \rfloor = \lceil n/2 \rceil$.
Since n is even or n is odd, it follows that $\lfloor (n + 1)/2 \rfloor = \lceil n/2 \rceil$.

Specialization: If c∈D and ∀x∈D.a(x) is true, then a(c) is true.

If c∈D, then (∀x∈D.a(x)) IMPLIES a(c) is a tautology.
If ∀x∈D.a(x) is an axiom,
then a(c) follows by modus ponens.

Generalization:

Let x∈D
⋮
p(x)
Since x is an arbitrary element of D,
∀x∈D.p(x)

LEMMA: For all integers x, if x is even, then x2 is even.
Proof:
Let x be an integer,
Assume x is even.
Then x=2k for some integer k,
so x2=(2k)2=2×(2k2), which is even.
Hence, x is even implies x2 is even.
Since x is an arbitrary integer,
for all integers x, if x is even, then x2 is even.

Construction:

Let x = __
⋮
Therefore x∈D
⋮
p(x)
∃x∈D.p(x)

Instantiation:

∃x∈D.p(x)
Let y∈D be such that p(y)
⋮

For any integer y, let largest(y)=NOT(∃x∈Z.(x>y))
Theorem: NOT(∃y∈Z.largest(y))
Proof:
1.Assume ∃y∈Z.largest(y)
2.Let y be an integer that satisfies largest(y). instantiation 1
3.NOT(∃x∈Z.x>y) definition 2
4.Let x=y+1
5.∀z∈Z.(z+1 is an integer) axiom.
6.x=y+1 is an integer. specialization 5
7.∀z∈Z.(z+1>z) axiom
8.x=y+1>y specialization 7
9.∀x∈Z.(x>y) construction 6,8
10.NOT(∃y∈Z.largest(y)) proof by contradiction 1,3,9

Notes about Indentation (indent for clarify notion of scope, some variables only defined in the indented area)
indent — ①Let .. when we pick/choose something, for generalization, instantiation, specialization, construction)
②Assume .. make an assumption ③Case write a tautology first

Induction

Simple Induction

Let p: $\mathbb{N}$→{T, F} be a predicate ∀n∈$\mathbb{N}$.p(n)
a.p(0) base case, basis step
b.Let n∈N be arbitrary.
c.Assume p(n)
x.p(n+1)
y.p(n) IMPLIES p(n+1) direct proof c, x
z.∀n∈N.[p(n) IMPLIES p(n+1)] generalization b, y (y, z optional)
∀n∈N.p(n) Induction a, z OR a, b, c, x

Theorem: consider any square chessboard whose sides have length which is a power of 2. If any one square is removed, then the resulting shape can be tiled using only 3-square L-shaped tiles.

Proof: for all n∈N, Let P(n)='any $2^n \times 2^n$ chessboard with 1 square removed can be tiled using 3-square L-shaped tiles'
Let Cn=set of all 2n×2n chessboard with 1 square removed.
Let 'L-tile' denote a 3-square L-shaped tile. ☆
P(n)='∀c∈C_n, (c can be tiled using only L-tiles)'
∀n∈N, P(n)
Base Case: P(0) is true C_0={■}
A $2^0 \times 2^0$ chessboard with 1 square removed has no squares and hence, can be tiled with 0 tile.
Let n∈N be arbitrary.
Suppose P(n) (is true)
Let c∈C_{n+1} be arbitrary.
Divide c into 4 equal $2^n \times 2^n$ chessboard. One of these has a square removed, so it is in C_n, hence by IH, it can be tiled with L-tiles.
Consider the other 3 cheeseboards. Each has 1 square that is one of the 4 squares in the middle of c. With these square removed, they are in C_n, so IH implies that they can be tiled by L-tiles. The 3 squares in the middle can be tiled with 1 L-tile.
c can be tiled using L-tiles.
∀c∈C_{n+1}, c can be tiled using L-tiles.
P(n+1) generalization Note: a universal quantification can be proved by induction or generalization

Theorem 2: All square chessboards with sides of length a power of 2 and with 1 square removed from the middle can be tiled using L-tiles (special case of theorem 1)

Theorem: ∀n∈N, $2n+1 \leq 2^n$

Proof: For n∈N, Let q(n)='$2n+1 \leq 2^n$' ☆Define Predicate!!!
Base Case: $2 \cdot 0+1=1=2^0$, so q(0) WRONG! False, for n=1 n=2
Note: look at small cases, figure out where 'it' starts~~

∀n∈N, n≥3 IMPLIES $2n+1 \leq 2^n$ Let M={n∈N | n≥3}, ∀n∈M, q(n)
Idea: Let p(n)=q(n+3) for all n∈N, then ∀n∈N, p(n) means the same as ∀n∈M, q(n)

∀n∈N, p(n)
Basis: p(0) is true
Inductive Step: Let n∈N be arbitrary.
Assume p(n)
…
p(n+1)
∀n∈N, p(n)

∀n∈M, q(n)
Basis: q(3)
Inductive Step: Let n∈N be arbitrary.
Assume q(n)
…
q(n+1)

Another way: Let r(n)='n≥3 IMPLIES q(n)'
∀n∈N, p(n) means the same as ∀n∈M, (n≥3 IMPLIES q(n))
r(0), r(1), r(2) are vacuously true.
Let n∈N be arbitrary.
Assume r(n),
Assume n+1≥3
..
r(n+1)

To prove ∀n∈N.[n≥b IMPLIES p(n)], it suffices to prove p(b) and ∀n∈N.(n≥b AND p(n) IMPLIES p(n+1))

Prove q(n) is true for all even natural numbers. ※
Let p(k)=q(2k),∀k∈N.p(k) means the same as ∀k∈N.q(2k), which is the same as ∀n∈N.(n is even IMPLIES q(n))
Base Case: p(0)=q(0)
Induction Step: p(k) IMPLIES p(k+1) which is the same as q(2k) IMPLIES q(2k+2)
It is sufficient to prove q(0) and ∀n∈N.(q(n) IMPLIES q(n+2))
—might be false since even if "q(n) is true for all even natural numbers", q(1)=T and q(3)=F is possible

Complete Induction

To Prove ∀i∈{0, ..n}. p(i)
Base Case: p(0)
Inductive Step: Let i∈{0,..n-1} be arbitrary (Not n since p(n+1) might be false)

Assume p(i)
p(i+1)
∀i∈{0,..n-1}.[P(i) IMPLIES P(i+1)] direct proof & generalization
Let i∈{0,..n}. P(i) Induction

Strong Induction: To prove ∀i∈N. p(i), it suffices to prove ∀i∈N.[∀*j∈N. j<i IMPLIES P(j)*] IMPLIES P(i)
Let i∈N be arbitrary.
Assume ∀j∈N.
∀*j∈N. j<i IMPLIES P(j)*
… (various cases)
P(i)
∀i∈N.[∀*j∈N. j<i IMPLIES P(j)*] *IMPLIES* P(i) direct proof + generalization
∀i∈N. P(i) Strong Induction

Theorem: ∀n∈Z+ and all a1, .. an∈R+, $\left(\prod_{i=1}^{n} a_i\right)^{\frac{1}{n}} \le \frac{\sum_{i=1}^{n} a_i}{n}$ (Geometric Mean ≤ Arithmetic Mean)

Proof: Let P(n)="for all a1, .. an∈R+, $\left(\prod_{i=1}^{n} a_i\right)^{\frac{1}{n}} \le \frac{\sum_{i=1}^{n} a_i}{n}$" We will prove ∀n∈Z+.P(n)

Base Case: n=2.
Let a1, a2∈R+ be arbitrary.
Then 0≤(a1-a2)2=a12-2a1a2+a22 so a12+a22≥2a1a2

Theorem: ∀n≥4, exactly can be made using only two Tonnies & 5$ bills.

Proof: Let n∈N be arbitrary.
Suppose n≥4 and ∀*j∈N. [4≤j<n IMPLIES P(j)]*
If n=4, then n=2×2+0×5
If n=5, then n=0×2+1×5
If n≥6, then 4≤n-2<n
P(n-2) is true by specialization
so ∃f∈N, ∃g∈N, n-2=2f+5g, so n=3f+5g
Hence P(n)
∀n∈N, P(n) Strong Induction

Theorem: Every integer greater than 1 is a product of primes (1 or more prime numbers)
Proof: For n∈N, Let P(n)='n is a product of primes'.
Let n∈N be arbitrary.
Suppose n>1 and ∀i∈N, [1<i<n IMPLIES P(i)]
If n is a prime, then n is a product of 1 prime, so P(n).
Otherwise, there are positive integers 1<k,m<n such that n=k·m.
By IH, k and m are products of primes, i.e. P(k) and P(m) are true. Thus, n=k·m is a product of primes.
($n = \prod_{i=1}^{k} p_i$ where k∈Z+ and p1, ..pk are primes k can be 1)
∀n∈N, [(n>1) IMPLIES P(n)]

Strong Induction from Weak Induction

(Weak) Induction inference rule says that from P(0) and ∀n∈N.(P(n) IMPLIES P(n+1)), one can infer ∀n∈N.P(n)
Strong Induction inference rule says that from ∀i∈N.(∀j∈N.[j<i IMPLIES Q(j)] IMPLIES Q(i)), one can infer ∀i∈N.Q(i)
Prove Strong Induction inference rule using (Weak) Induction:
If ∀i∈N.(∀j∈N.[j<i IMPLIES Q(j)] IMPLIES Q(i)) is an assumption (a temporary axiom),
then we can prove ∀i∈N.Q(i).

Define R: N→{T, F} so that for each i∈N, R(i)=∀j∈N.(j<i IMPLIES Q(j))

1. ∀i∈N.([∀j∈N.(j < i implies Q(j))] implies Q(i)) assumption

Recursively-Defined Set

These definition has 2 parts: a base case, that doesn't depend on anything else.
a construction case, that depends on previous cases.

{0, 1}*= set of all finite strings of bits
—base case: λ, empty string of length 0 is in {0, 1}*
—constructor case: if s∈{0, 1}*, then s0 and s1 are in {0, 1}*
For any set ∑, ∑* is the set of all finite length strings of letters from ∑

Brkts = set of finite strings of matched brackets
—base case: λ∈Brkts
—constructor case: if s∈Brkts, then [s]∈Brkts, s[]∈Brkts. ✘ WRONG! cannot build [[]] [[]]
if s, t∈Brkts, then s[t]∈Brkts
if s, t∈Brkts, then st∈Brkts, [s]∈Brkts
S=syntactically correct formulas of propositional logic.
—base case: propositional variables as in S (Let P be a variable, P∈S)
—constructor case: if f, g∈S, then NOT(f)∈S, (f AND g)∈S, (f OR g)∈S… (f*g∈S, where * is a connective)
M=syntactically correct monotone formulas of propositional logic
—base case: propositional variables in M
—constructor case: if f, g∈M, then (f OR g), (f AND g)∈M, Note that NOT(f)∉M
S=set of formula where negative only appears at literals NOT(f*g)=f※g (can be replaced with)
M=the smallest set of formulas containing all the propositional variables and closed under AND and OR.

Structural Induction (prove properties above recursively defined sets)

Recursively Defined Predicate P: S→{T, F}
To prove ∀s∈S, P(s) by structural induction, where P: S→{T, F} is a predicate
Prove P(s) is true for all base cases s of the definition S.
P(s) for the constructor cases s of the definition, assuming P is true for the components of S.

For all f∈M, let Npv(f)=# of occurrences of propositional variables in f, Nop(f)=# of occurrences of binary connectives in f.
Predicate: P(f)='Npv(f)=1+Nop(f)'
—base case: if f is a propositional variable, then Npv(f)=1 and Nop(f)=0, so P(f) os true.
—constructor case: consider f=(f' OR f'') f=(f' c f'') where c is either AND or OR
Assume P(f') and P(f'')
Npv(f)=Npv(f')+Npv(f'') Nop(f)=Nop(f')+Nop(f'')+1
By IH, Npv(f')=Nop(f')+1 and Npv(f'')=Nop(f'')+1
Npv(f)=Npv(f')+Npv(f'')=Nop(f')+1 + Nop(f'')+1 = [Nop(f')+ Nop(f'')+1]+1=Nop(f)+1, so P(f)
Similarly, if f=(f' AND f''), P(f) is true.
By structural induction, ∀f∈M, P(f).

N can be defined recursively, base case: 0∈N; constructor cases: if n∈N, then (n+1)∈N
N×N can be defined recursively, base case: (0, 0)∈N×N; constructor cases: if (m, n)∈N×N, then (m+1, n), (m, n+1)∈N×N ((m+1, n+1) is redundant, can be built from (m+1, n) then (m, n+1))

To Prove ∀m∈N, ∀n∈N, P(m, n) (P: N×N→{T, F})
Let m∈N be arbitrary.
Prove ∀n∈N, P(m, n) by induction or generalization
∀m∈N, ∀n∈N, P(m, n) by generalization

Assume P(i, j) for all(i, j) where i≤m and j≤n and either i<m or j<n and prove P(m, n).
(strong induction version of structural induction, meaning i≤m and j<n OR i<m and j≤n, then prove P(m, n))

—can define functions on recursively defined sets

for f∈M, let n(f)='the number of occurrences of propositional variables in f'
n(P)=1, for any propositional variable P
n(f)=n(f')+n(f'') for f=(f' OR f'') and f=(f' AND f'')

—Let B be the set of al binary trees

Base Case: empty tree is in B Constructor Case: if t1, t2∈B and r is a node then  ∈B

—For l∈B, let N(l)=# of nodes in l we say t1=left(t), t2=right(t)
Base Case: N(empty tree)=0 Constructor Case: N(t)=1+N(left(t))+N(right(t))

—L(t)=# of leaves of t
Base Case: L(empty tree)=0, L(one node tree)=1
Constructor CaseL L(t)=L(left(t))+L(right(t))

Theorem: a binary tree with n nodes has at most ⌈n/2⌉ nodes.

①∀t∈B, L(t)≤⌈N(t)/2⌉ structural induction

②∀n∈N, ∀t∈B, N(t)=IMPLIES L(t)≤⌈n/2⌉ (if omit "∀n∈N", n is undefined) strong induction on variable n

③For t∈B and n∈N, let S(t, n)='t has n nodes' and A(t, n)='t has at most n leaves'.
∀n∈N, ∀t∈B, (S(t, n) IMPLIES A(t,⌈n/2⌉))

Define Predicate: Let P(n)='∀t∈B, S(t, n) IMPLIES A(t,⌈n/2⌉)'
Let n∈N be arbitrary.
Suppose ∀i∈N, (i<n IMPLIES P(i)) Induction Assumption (since N natural number, no need to talk about i>0 explicitly)
Let t∈B be arbitrary.
Suppose S(t, n)
…
A(t,⌈n/2⌉)
S(t, n) IMPLIES A(t,⌈n/2⌉) direct proof
∀t∈B, S(t, n) IMPLIES A(t,⌈n/2⌉) generalization
P(n) generalization
∀n∈N, P(n) strong induction

To Prove A(t,⌈n/2⌉): proof by cases
Case 1: n=0, then t has 0 nodes and 0 leaves, since 0=⌈0/2⌉, A(t, 0) is true Case 2: n=1
Case 3: n>1 (不能用n>0, because defn of L(t), BE CAREFUL~)
Then t has a root, a left subtree t' and a right subtree t''
Let n'=N(t'), S(t', n') true, n''=N(t''), S(t'', n'') true, n=n'+n''+1, so n', n''<n
By IH & specialization, A(t',⌈n'/2⌉), A(t'',⌈n''/2⌉) are true
L(t)=L(t')+L(t'')≤⌈n'/2⌉+⌈n''/2⌉≤(n'+1)/2 + (n''+1)/2 =(n+1)/2 = ⌈n/2⌉ since L(t)∈N
※Since L(t)∈N, integer, if n is even (n+1)/2=n/2+1/2 ⌈n/2⌉=n/2 if n is odd, (n+1)/2 = ⌈n/2⌉ ???
Thus L(t)≤⌈n/2⌉ since L(t)∈N (Let q(t)='L(t)≤⌈n/2⌉' ∀t∈B, q(t))
Since L(t)≤n/2+1/2 and L∈N, L(t)≤n/2=⌈n/2⌉

Well-Ordering Principle

An order set (or partially ordered set) S is **well-ordered** if every non-empty subset of S has a smallest element.

N ✓ Z ✘ Q ✘

Z ordered by absolute value, then by value: 0, -1, 1, -2, 2…
Q+ considered in reduced form, ordered first by denominator then numerator: 1, 2, 3 .. 1/2, 3/2, 5/2, .. 1/3, 2/3, 4/3…
ordered by max{numerator, denominator} when written in reduced form, then by value: 1/1, 1/2, 2/1, 1/3, 2/3, 3/2…

∀e∈S, P(e) can be proved using the well-ordering principle for any well-ordered set S with ordering α.

L1 To obtain a Contradiction, Suppose that ∀e∈S, P(e) is False.
L2 Let C={e∈S | P(e) is False} be the set of counterexamples to P
L3 C≠∅ by definition, L1 L2
L4 Let e be the smallest element of C, well ordering principle L3
Let e'=…

L5 e'∈C

…

L6 e'<e

L7 This is a contradiction L4,L5,L6

L8 ∀e∈S, P(e) Proof by contradiction L1, L7

Let P(n)='n can be written as a product of primes'

Let C={n∈N | P(n) is false}

Assume (∀n∈N, P(n)) is false

C≠∅

Then by well-ordering principle, C has a smallest element

Theorem: Every integer greater than 1 can be written as a product of primes.

Proof: Suppose the claim is False.

Let n be the smallest integer that cannot be written as a product of primes.

Then n is not prime since a prime is the trivial product of itself. Therefore, n is composite.

So there exist k, m∈N such that k>1 and m>1 and n=k·m

But k, m are smaller than n so they can be written as product of primes. Hence n is a product of primes (Contradiction)

Theorem: Every positive natural number m/n can be expressed in reduced form m'/n' where m' and n' have no common factors.

Proof: P(m, n)='natural number m/n can be expressed in reduced form'

∃m, n such that P(m, n)=F

C={m∈Z+ | ∃n∈Z+, such that P(m, n)=F}

By well-ordering principle, let m0 be the smallest element in C.

∃n0∈Z+ such that P(m, n)=F

Since m0/n0 is not in reduced form, there is prime divisor p≥2

Let m'=m0/p, n'=n0/p, we know that P(m', n')=T, since m'<m0

However, m0/n0=m'/n' which means m0/n0 can be expressed in reduced form. This is a contradiction

∀m, n such that P(m, n)=F

Sets & Diagonalization

A function f: D→R is **onto/surjective** if ∀y∈R.∃x∈D.f(x)=y

From the existence of such a function, if D and R are finite sets, then |D|≥|R|

(D has to be at least big as R, but D can be larger)

A set C is **countable** if and only if C=∅ or there is a surjective function from $\mathbb{N}$ to C.

A nonempty set C is countable if there is a surjective function from N to C.

Every finite set is countable. Empty set is countable.

Let C={c_1, …c_n} be a finite non-empty set, Define f: $\mathbb{N}$→C to be the function $f(i)=\begin{cases} c_{i+1} \ if \ i<n \\ c_n \ if \ i\geq n \end{cases}$

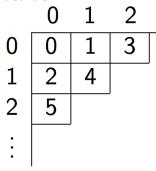
$\mathbb{Z}$ is countable. Define f: $\mathbb{N}$→$\mathbb{Z}$ to be the function $f(i)=\begin{cases} 0, \ if \ i=0 \\ j, \ if \ i=2j-1 \ for \ some \ j\in\mathbb{Z}^+ \\ -j, \ if \ i=2j \ for \ some \ j\in\mathbb{Z}^+ \end{cases}$ $f(x)=\begin{cases} \frac{-x+1}{2}, if \ x \ is \ odd \\ \frac{x}{2}, if \ x \ is \ even \end{cases}$

0, -1, 1, -2, 2…

0, 0, 1, -1, 2, -2…

If A and B are countable, then A∪B is countable. If A is countable and B⊆A, then B is countable.

{x∈Z| x is odd} is countable



If A and B are countable,
then A×B={(a, b) | a∈A AND b∈B} is countable.
ℕ×ℕ is countable. Z is countable, so Z×Z is countable.

If A is countable and there is a surjective function f: A→B, then B is countable.

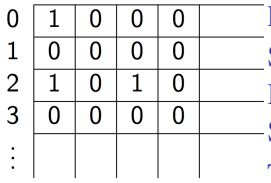
Proof: Since A is countable, there is a surjective function g: ℕ→A.
Let function h: ℕ→B be such that, for all i∈ℕ, h(i)=f(g(i)).
To prove that h is surjective, consider any Z∈B, since f is surjective, there exists y∈A such that f(y)=C. Since y is surjective, there exists x∈N such that g(x)=y.
For each z∈B, there exists y∈A such that f(y)=z and there exists x∈N such that g(x)=y. Hence h(x)=f(g(x))=f(y)=z.
Thus h is surjective and B is countable.

ℚ is countable. Q=Q≥0∪Q<0. Define f: ℕ×ℕ→ℚ to be the function $f(p,q)=\begin{cases} p/q, \text{ if } q\neq 0 \\ 0, \text{ if } q=0 \end{cases}$

The set of all finite binary sequences is countable: ε, 0, 1, 00, 01, 10, 11, 000 …
The set of all finite strings of ASCII characters is countable.
The set of all syntactically correct Python programs is countable.

For any set A, power set of A is the set of all subsets of A, P(A)={S|S⊆A}. If |A|=n, then |P(A)|=2^n.

Theorem: P(ℕ) is uncountable



Proof: Suppose P(ℕ) is countable. Then there exists a surjective function f: ℕ→P(ℕ).
Let D={i∈ℕ | i∉f(i)}⊆ℕ, i.e D∈P(ℕ), since P(ℕ)={S | S⊆ℕ}
Since f is surjective, there exists j∈N such that f(j)=D. Then for all i∈ℕ, i∈f(j) IFF i∈D, since f(j)=D.
By definition of D, i∈D IFF i∉f(i).
Since j∈N, by specialization j∈f(j) IFF j∉f(j). This is a contradiction.
Therefore, P(ℕ) is uncountable.

Proof by Diagonalization: M[i, j]=1 IFF j∈f(i)
Each row is a characteristic vector of a set (Each row indicates one subset of A)
Since f is surjective, every subset of N is represented by some row.
Since f is not necessarily injective, the same set can be represented by multiple rows.
i∈D IFF M[i, j]=0. This ensures that D≠f(i).

S⊆{1, 2, 3, 4} can be represented by a binary sequence S1, S2, S3, S4 where $S_i=\begin{cases} 1, if\ i\in S \\ 0, if\ i\notin S \end{cases}$

Characteristic Vector of set S: 0,1,1,0 denotes {2, 3} 0,0,0,0 denotes ∅

If S⊆N the characteristic vector is an infinite binary sequence S0, S1, S2… where $S_i=\begin{cases} 1, if\ i\in S \\ 0, if\ i\notin S \end{cases}$

f(0) $f(0)_0\ f(1)_1\ f(2)_2$ …
f(1) $f(1)_0\ f(1)_1\ f(2)_2$ …
f(2) …
f(i) list of all subsets of N possibly with duplications

This is an infinite 2-dimensional Boolean array M where $M[i, j]=f(i)_j$ where i is row j is column.
The characteristic vector of D is the complement of the diagonal of the matrix M.
The characteristic vector of D does not agree with row i of M in column i.
M[i, i]=1 IFF i∈f(i)
characteristic vector=1 IFF i∈D IFF i∉f(i)

Thus
f(0)

f(1)
..

does not contain D
This contradicts the fact that f is surjective.

Let F be the set of all functions from $\mathbb{N}$ to $\mathbb{N}$, F is uncountable.

Proof: Suppose F is countable
Then there is a surjective function f: N→F
f_i is a function from $\mathbb{N}$ to $\mathbb{N}$, create an infinite 2-dimensional matrix where row i corresponds to $f_i \in F$

	0	1	2	3	… j	
f_0	$f_0(0)$	$f_0(1)$	$f_0(2)$	$f_0(3)$		function f_0 value on 0, 1, 2… ($\mathbb{N}$)
f_1	$f_1(0)$	$f_1(1)$	$f_1(2)$	$f_1(3)$		
f_2	…		$f_2(2)$			
f_3						

define $g(n)=f_n(n)+1$, $g \in F$ (increase the element at diagonal by 1)
Assume that $g=f_n$ for some $n \in N$, Then $g(n)=f_n(n)$
By definition, $g(n)=f_n(n)+1 \neq f_n(n)$ Contradiction.

If ∑ is a finite set of letters, then ∑* is the set of all finite strings of letters from ∑.

For program P∈ ASCII*, input x∈ ASCII* (programs/functions are ASCII strings)
Let H(P, x)=1, if P returns/halts on input x and P is a syntactically correct function with 1 input; and 0 otherwise.

The halting problem is solvable if such a C function H exists. (this C function always returns 0 or 1)

Theorem: the halting problem is unsolvable

Proof: To obtain a contradiction, suppose that such a C function H exists.
Consider the syntactically correct C function D (D is another C function)

```
D(x):                    # D is a syntactically correct C function
{
If H(x, x)
while (1);
else return;
}
```

when D runs on input D,
If H(D, D)=0, then D returns on input D. If H(D, D)=1, then D goes into an infinite loop on input D.
From the definition of H, if D returns on input D, then H(D, D)=1
If D goes into an infinite loop on input D, then H(D, D)=0. Contradiction!

Correctness and Analysis of Iterative and Recursive Algorithms

Readings:
Mathematics for Computer Science, chapter 22
236/240 course notes chapters 2, 3
Introduction to Algorithms, chapters 3, 4

Let F denote the set of all functions from natural number N to nonnegative reals $R^{\geq 0}$.
For any $f \in F$, let $O(f)=\{g \in F \mid \exists c \in R^+, \exists b \in N, \forall n \in N, (n>b \text{ IMPLIES } g(n) \leq c \cdot f(n))\}$

$\Omega(f)=\{g \in F \mid \exists c \in R^+. \exists b \in N. \forall n \in N. (n \geq b \text{ IMPLIES } g(n) \geq cf(n))\}$

$\Theta(f)=\{g \in F \mid \exists c_1 \in R^+. \exists c_2 \in R^+. \exists b \in N. \forall n \in N. (n \geq b \text{ IMPLIES } c_1(n) \leq g(n) \leq c_2 f(n))\}=O(f) \cap \Omega(f)$

Analysis of Algorithm

For an algorithm A, let $t_A(T)$ = number of steps algorithm A takes on input I.
What is a step?
pick 1 or 2 operations such that total number of operations performed by A is the same as the number of these operations performed by A, to within a constant factor.

Properties of O Notation: (Similar properties hold for Ω and Θ)
①constant factors don't matter: if d>0 is a constant, then df(n)∈O(f(n)), f(n)∈O(df(n)). (df is in O(f) and f(n) is in O(df))
②lower order terms don't matter: if $\lim_{n\to\infty}\frac{h(n)}{g(n)}=0$, then g(n)+h(n)∈O(g(n)). g(n)+h(n)≤g(n)+c·g(n)=(1+c)g(n)∈O(g(n))
③Transitivity: if f(n)∈O(g(n)) and g(n)∈O(h(n)), then f(n)∈O(h(n))
④Summation Rule: if f1∈O(g1), f2∈O(g2), then f1+f2∈O(g1+g2)
⑤Product Rule: if f1∈O(g1) and f2∈O(g2) then f1×f2∈O(g1×g2)
⑥max{f, g}∈O(f+g), f+g∈O(max{f, g})
⑦if f∈O(g), then max{f, g}∈O(g)
⑧Exponents & Logarithm: let a, b be constants (a, b∈R)
Exponents matter, if a≤b then $n^a\in O(n^b)$ but $n^b\notin O(n^a)$
Bases of exponents matter, if 1<a≤b, then $a^n\in O(b^n)$ but $b^n\notin O(a^n)$
Bases of logarithms don't matter, for all a, b >1, $\log_a(n)\in O(\log_b(n))$ (consider change of bases)
Exponential functions grow faster than polynomial functions for all b>1 and all a, $n^a\in O(b^n)$ but $b^n\notin O(n^a)$
Polynomial functions grow faster than poly-logarithmic functions, for all a, b>0, c>1, $(\log_c n)^a\in O(n^b)$ but $n^b\notin O((\log n)^a)$

```
LS(L, x):  linear search
% if x occurs in L, return index of L at which x occurs, otherwise return 0. L is an array with first
index 1.
i←1
while i≤length(L) do
     if L[i]=x
     then return i
     i←i+1
end while
return 0
```

count number of comparisons with x ☆define what I am counting clearly
each iteration of the loop performs O(1) steps, assuming length takes O(1) step
outside the loop, O(1) step are performed.

On Input		# comparisons with x
L=[2, 4, 6, 8]	x=2	1
	x=4	2
	x=8	4
	x=1	4

express complexity as a function of the input size

max{$E_{tA}(I)$} worst-case complexity (need input size)
T_A: N→N $T_A(n)=\max\{E_{tA}(I) \mid size(I)=n\}$ worst-case time complexity of algorithm A
For LS, size((L, x))=length(L)

Average-Case time complexity
T_A': N→$R^{\geq 0}$, where $T_A'=E[t_A]$, the expectation is taken over a probability space of all inputs of size n
If all inputs of size n are equally likely, $T_A'(n)=\frac{\sum\left(t_A(I)\middle| size(I)=n\right)}{\#\left\{I\middle| size(I)=n\right\}}$

Consider L∈{0, 1}^{n} and x=1,

L[i]=0 means L[i]≠x, L[i]=1 means L[i]=x
If we are restricting to sorted list, could use L=[1, 2, ..n] x∈{..n}
L=[2, 4, ..2n] x∈{1, 2, 3..,2n+1}

Languages and Automata Theory

Quiz 2

Consider the predicate logic formula ∃f∈F. ∀x∈D. ∀y∈D. [p(f(x), y) IMPLIES p(f(y), x)].
Give an interpretation that makes this formula true.
Let D={1}, let F={f}, where f: D → D maps 1 to 1, and let p: D×D→{T, F} be the predicate that maps (1,1) to T.
Give an interpretation that makes this formula false.
Let D={1, 2}, let F ={f}, where f: D → D is the identity function, and let p: D×D→{T, F} be the < predicate.

Quiz 4

Prove by induction for all natural numbers n, n^3+2n is a multiple of 3.
For n∈N, let predicate P: N→{T, F} be the predicate such that P(n)="n^3+2n is a multiple of 3" or "∃m∈N, ($n^3+2n=3\cdot m$)"
Base Case: n=0, $0^3+2\times0=0=3\times0$, so P(0) holds by construction.
Let n∈N be arbitrary and suppose P(n) is true.
Then, there exists m∈N such that $n^3+2n=3\cdot m$ by instantiation.
Since $(n+1)^3+2\times(n+1)=n^3+3n^2+3n+1+2n+2=(n^3+2)+3(n^2+n+1)=3(m+n^2+n+1)$, it follows that P(n+1) is true by construction.
P(n) IMPLIES P(n+1) by direct proof. ∀n∈N, P(n) IMPLIES P(n+1) by generalization.
Conclude that ∀n∈N, P(n) by induction.

Quiz 5

Consider a game in which there are 2 piles of coins and 2 players. Players alternately take turns. In each turn, a player must remove 1 or more coins from one of the 2 piles of coins. The player who removes the last coin is the winner. Using the well-ordering principle, prove that if the two piles are initially nonempty and contain the same number of coins, the second player can play so that she will win, no matter what the first player does.

Say that a player has a winning strategy if that player can play so that she will win, no matter what the first player does.
Let predicate P : Z+→{T, F} be the predicate such that P(n)="if the two piles initially contain n coins each, the second player has a winning strategy"
Let C={n∈Z+ | P(n) is false}.
To obtain a contradiction, suppose that C≠∅.
Let n be the smallest element of C, which exists by the well ordering principle.
Consider any turn by the first player, when each pile initially contains n coins.
Say the first player removes i coins from one pile. If i=n, the second player can win by removing all n coins from the other pile.
So, suppose that i<n.
In this case, the second player removes i coins from the other pile. Now each pile contains n−i coins.
Since n−i<n, the definition of n implies that n−i∉C.
Thus, from this point, the second player has a winning strategy.
Hence, by generalization, no matter how many coins the first player takes in his first step, the second player can play so that she will win.
This is a contradiction, since n∈C.
Therefore, C=∅ and we have proved ∀n∈Z+. P(n)

Quiz 6

Consider a robot that moves on the infinite plane, starting at (0,0). Each step, it moves distance 1, either left, right, up, or down. Let L denote the set of possible infinite paths that the robot could take.
One example of such a path is when the robot repeatedly moves distance 1 to the right, getting increasingly far from the origin.
Use diagonalization to prove that L is uncountable.

Each path $\pi \in L$ can be viewed as an infinite sequence $\pi_0, \pi_1 ..$ each of whose elements π_i, for $i \in N$ is one of Left, Right, Up, or Down.

To obtain a contradiction, suppose that L is countable. Then there is a surjective function f: N→L.

Consider the path δ where for $i \in N$, step i is $\delta_i = \begin{cases} Right, & if\ f(i)_i \in \{Left, Up, Down\} \\ Left, & if\ f(i)_i = Right \end{cases}$

Note that $\delta \in L$, since it is an infinite sequence each of whose elements is one of Left, Right, Up, or Down.

Since f is surjective, there exists $j \in N$ such that $f(j)=\delta$. But $\delta_j \neq f(j)_j$, so $f(j) \neq \delta$. This is a contradiction. Hence L is uncountable.

Alternative:

To obtain a contradiction, suppose that L is countable.

Then there is a surjective function f: N→L.

Consider function g: L→P(N) such that for each path $\pi \in L$, $g(\pi)=\{i \in N \mid$ the i-th step of π is Right$\}$

Note that g is surjective, since for each $S \in P(N)$, there is a sequence $\pi \in L$ such that $g(\pi)=S$, namely the sequence π such that i-th step of π is Right if $i \in S$ and i-th step of π is Left if $i \notin S$.

Since the composition of surjective functions is surjective, f∘g: N→P(N) is surjective. Thus P(N) is countable.

But, we used diagonalization to prove that P(N) is uncountable. Hence L is uncountable.

Problem Session 1

Problem 1.1: I wont use Uber unless the city has created regulations

define variable, let U denote "I will use Uber", let C denote "the city has created regulations"

(NOT U) UNLESS C

If the city hasn't created regulations, I wont use Uber. (NOT C) IMPLIES (NOT U) = U IMPLIES C

(NOT P) UNLESS Q means P IMPLIES Q, or P IFF Q.

P UNLESS Q means (NOT P) IMPLIES Q, or (NOT P) IFF Q

Problem Session 2

Problem 2.1:

$\forall x \in N, \exists y \in N, (2x-y=0)$ True (given $x \in N$, let $y=2x$)

$\exists y \in N, \forall x \in N, (2x-y=0)$ False (given $y \in N$, let $x=y+1$, then $2x-y=y+2>0$)

$\forall x \in N,$ (x<10 IMPLIES $\forall y \in N,$ (y<x IMPLIES y<9)) True (If x<10 then x≤9, hence if y<x, then y<9)

Problem 2.2:

a predicate floor: $R \times Z \to \{T, F\}$ such that for $x \in R$ and $y \in Z$,

floor(x, y)=T if y is the largest integer less than or equal to x

round(x, y)=T if y is the closest integer to x

Note: in definitions, the English word "if" often means "if and only if".

For all $x \in R$ and $y \in Z$,

floor(x, y) = "$(y \leq x)$ AND $(\forall z \in Z, (z \leq x)$ Implies $(z \leq y)$" = "$(y \leq x)$ AND $(x < y+1)$"

round(x, y) = "$\forall z \in Z, |z-x| \geq |y-x|$" = "$|y-x| < 0.5$"

Note that both round(0.5,1)=T and round(0.5, 0)=T. Modify this predicate so that if x=y+0.5 for some integer y, then round(x, y+1)=T but round(x, y)=F

Problem 2.3:

Let U be a set of elements, let Member: $U \times P(U) \to \{T,F\}$ be the predicate such that Member(x, S)="element x is in set S".

Predicate denotes “intersection of subsets A and B is empty”

For all subsets A and B of U, let EmptyIntersection(A,B) = “NOT[$\exists x \in U.(M(x,A)$ AND $M(x,B))]$” = “$\forall x \in A.$ NOT(M(x,B))”=“$\forall x \in B.$ NOT(M(x,A))”

Problem 2.4:

t(x, y, z) = “process x transmitted process y’s message to process z”, and

w(x, y) = “process x wants to send a message to process y”.

q denotes a particular process. Let P denote the sets of processes.

(a)No process transmitted q’s message, but some process wants to send a message to q.

NOT($\exists x \in P. \exists z \in P.$ t(x, q, z)) AND ($\exists y \in P,$ w(y, q)) = ($\forall x \in P. \forall z \in P.$ NOT(t(x, q, z))) AND ($\exists y \in P,$ w(y, q))

(b)Every process transmitted its message to a process that does not want to send it a message.

$\forall x \in P. \exists z \in P.$[t(x, x, z) AND NOT(w(z, x))]

(c)Some process wants to send a message to every process that transmitted its message.

$\exists x \in P,$ [$\forall y \in P, \exists z \in P,$ w(x, y) AND t(y, x, z)]

Right Answer: $\exists y \in P. \forall x \in P. \forall z \in P.$ [t(x, y, z) IMPLIES w(y, x)] = $\exists y \in P. \forall x \in P.$[($\exists z \in P.$ t(x, y, z)) IMPLIES w(y, x)]

Or $\exists y \in P. \forall x \in P. \forall z \in P.$ [t(x, x, z) IMPLIES w(y, x)] = $\exists y \in P. \forall x \in P.$[($\exists z \in P.$ t(x, x, z)) IMPLIES w(y, x)]

Tutorial 1

Tutorial 1: Consider the following piece of code from a distributed algorithm that is being performed by n processors, p_1,.., p_n. All variables in the program are of type integer.

```
1.if epoch>e
2.   then return
3.if counter=0 or epoch<e
4.      then temp ← 1
5.      else temp ← temp + 1
6.epoch←e
7.if epoch≤e
8.      then counter ← temp
9.return
```

per(p, l) = “processor p has performed line l”.

input(p, v) = “integer v is the value of the input to processor p”.

value(x, v) = “integer v is the value of variable x”.

Translate the following sentences from the proof of correctness of the algorithm:

Let P = {$p_1, \ldots, p_n$} denote the set of processors. ☆

①If some processor has returned, then counter≠0.

($\exists p \in P.$[per(p, 2) OR per(p, 9)]) IMPLIES NOT value(counter,0)

②Whenever a processor with input e has performed line 6, epoch has value at least e.

$\forall e \in Z,$ [$\exists p \in P,$ (input(p, e) AND per(p, 6)) IMPLIES ($\exists f \in Z,$ value(epoch, f) AND $f \geq e$)]

③Some process has an input that has the same value as some variable that is not an input.

Let V={counter, epoch, temp} denote the set of variables that aren’t inputs.

$\exists p \in P, \exists x \in V, \exists e \in Z,$ [input(p, e) AND value(x, e)]

④Some later line has been performed by every process that has performed line 3.

Let L = {1,...,9} denote the set of lines.

$\exists l \in L,$ [(l>3) AND $\forall p \in P,$ (per(p, 3) IMPLIES per(p, l))]

Alternative: $\forall p \in P.$ (per(p, 3) IMPLIES (per(p, 4) OR per(p, 5) OR per(p, 6) OR per(p, 7) OR per(p, 8) OR per(p, 9)))

f: $R \to R$; a, $l \in R$; $(c, d) \subseteq R$

$\lim_{x \to a} f(x) = l$

$\forall \varepsilon \in R+, \exists \delta \in R+, \forall x \in R,$ ($0<|x-a|<\delta$ IMPLIES $|f(x)-l|<\varepsilon$)

f is differentiable in (c, d)

$\forall x \in (c, d), \exists l \in R, \forall \varepsilon \in R+, \exists \delta \in R+, \forall \Delta \in R,$ ($0<|\Delta|<\delta$ IMPLIES $|[f(x+\Delta)-f(x)]/\Delta - l|<\varepsilon$)

f, g: N→N $f \in O(g)$ $\exists c \in Z+. \exists b \in N. \forall n \in N.$ ($n \geq b$ IMPLIES $f(n) \leq c \cdot g(n)$)

Tutorial 2 TO DO

Tutorial 2: complete set of connectives

Problem Session 3

Problem 3.1:

(P IMPLIES Q) IMPLIES P Satisfiable, not valid (truth assignment P=Q=T makes it true, P=Q=F makes it false)

P IMPLIES (Q IMPLIES P) Valid (if P=F, then the formula is vacuously true. if P=T, then Q IMPLIES P is true)

Truth Table

P	Q	P IMPLIES Q	Q IMPLIES P	(P IMPLIES Q) IMPLIES P	P IMPLIES (Q IMPLIES P)
T	T	T	T	T	T
T	F	F	T	T	T
F	T	T	F	F	T
F	F	T	T	F	T

Problem 3.2: consider the truth table for Boolean predicate M(P, Q, R) which true when exactly two of P, Q, R are true

P	Q	R	M	DNF	NOT(M)	DNF for ~M
T	T	T	F		T	P AND Q AND R
T	T	F	T	P AND Q AND NOT(R)	F	
T	F	T	T	P AND NOT(Q) AND R	F	
T	F	F	F		T	P AND NOT(Q) AND NOT(R)
F	T	T	T	NOT(P) AND Q AND R	F	
F	T	F	F		T	NOT(P) AND Q AND NOT(R)
F	F	T	F		T	NOT(P) AND NOT(Q) AND R
F	F	F	F		T	NOT(P) AND NOT(Q) AND NOT(R)

Propositional formula in DNF for M(P,Q,R):

(P AND Q AND NOT(R)) OR (P AND NOT(Q) AND R) OR (NOT(P) AND Q AND R)

Propositional formula in CNF for M(P,Q,R):

NOT{(P AND Q AND R) OR (P AND NOT(Q) AND NOT(R)) OR (NOT(P) AND Q AND NOT(R)) OR (NOT(P) AND NOT(Q) AND R) OR(NOT(P) AND NOT(Q) AND NOT(R))}

=[NOT(P) OR NOT(Q) OR NOT(R)] AND [NOT(P) OR Q OR R] AND [P OR NOT(Q) OR R] AND [P OR Q OR NOT(R)] AND [P OR Q OR R]

=[NOT(P) OR NOT(Q) OR NOT(R)] AND

[NOT(P) OR Q OR R] AND [P OR Q OR R] AND

[P OR NOT(Q) OR R] AND [P OR Q OR R] AND

[P OR Q OR NOT(R)] AND [P OR Q OR R]

=[NOT(P) OR NOT(Q) OR NOT(R)] AND [Q OR R] AND [P OR R] AND [P OR Q]

At least two of P, Q, and R are true. + At least one of P, Q, and R is false.

Problem 3.3: Translate the following specifications into propositional formulas, using the four propositional variables: L = “file system is locked”, Q = “new messages are queued”, B = “new messages will be sent to the message buffer”, and N = “the system is functioning normally”.

a. If the file system is not locked, then

new messages are queued,

new messages will be sent to the message buffer,

system is functioning normally, and conversely, if system is functioning normally, then the file system is not locked.

b. If new messages are not queued, then they will be sent to the message buffer.

c. New messages will not be sent to the message buffer.

a.[NOT(L) IMPLIES (Q AND B AND N)] AND (N IMPLIES NOT(L))

b.NOT(Q) IMPLIES B

c.NOT(B)

This set of specifications is satisfiable, truth Assignment: B=F, Q=T, L=T, N=F

Explain why no other truth assignment satisfies these specifications:

To satisfy the third specification, we must have B = F.

To satisfy the second specification when B = F, we must have NOT(Q) = F, so Q = T.

To satisfy the first part of the first specification when B = F, we must have NOT(L) = F, so L = T. To satisfy the second part when NOT(L) = F, we must have N = F.

Problem Session 4

Problem 4.1: free occurrences of each variable in the predicate logic formula
[∀x ∈ D.∃y ∈ D.R(x,y,z) IMPLIES ∃w ∈ D.Q(w,x,y)] IMPLIES ∀y ∈ D.Q(w,w,y)

Question 4.2:

Translate the formula ∀x ∈ D.∀y ∈ D.[(G(y) AND ∀y ∈ D.(S(x,y) IMPLIES B(y))) IMPLIES NOT (S(y,x))] into an English sentence, using an interpretation where D is a nonempty set of people,
G(y) = "person y is a girl", B(y) = "person y is a boy" and S(x,y) = "person x is a sibling of person y".

Every girl is not the sibling of anyone whose siblings are all boys. Every girl is not the sister of anyone who only has brothers. Everyone who only has brothers has no sisters.
G(y) and S(y,x) are bound to the first ∀y ∈ D, S(x,y) and B(y) are bound to the second ∀y ∈ D
Note: If a variable is in the scope of multiple quantifiers, it is bound to the innermost quantifier applied to that variable.

Problem 4.3: Consider predicate logic formula ∀A∈ S.∀B∈ S.∀C∈ S.∀D∈ S.[(A×B ⊆ C×D) IMPLIES ((A⊆C) AND (B⊆D))], where × denotes the Cartesian product of sets and ⊆ denotes subset
—Interpretation which makes this formula True:
let S={{s}} consist of one set, which contains a single element s. The only choices for A, B, C, and D are A=B=C=D={s}. Since A×B={(s, s)}=C×D, both the hypothesis and the conclusion are true. Hence the implication is true. Therefore the formula is true under this interpretation.
—Interpretation which makes this formula False:
Let S={ϕ, {r}, {s}}. Let A=ϕ, B={r}, and C=D={s}.
Since A×B=ϕ ⊆{(s, s)}=C×D, the hypothesis is true. But B⊈D, so the conclusion is false. Hence the implication is false and the formula is false under this interpretation.

Problem 4.4: Translate the formula [∃x∈ N.(x=y)] IMPLIES ∃x∈ N.[x=0 OR NOT(∃y∈ N.(y<0))] into a logically equivalent formula in Prenex Normal Form. Use brackets where necessary to avoid ambiguity.
Substitute z for the second quantified x and w for the quantified y.
= [∃x∈ N.(x=y)] IMPLIES ∃z∈ N.[z=0 OR NOT(∃w∈ N.(w<0))]
Then apply a sequence of transformations to get Prenex Normal Form.
= [∃x∈ N.(x=y)] IMPLIES ∃z∈ N.[z=0 OR (∀w∈ N. NOT(w<0))] = [∃x∈ N. (x=y)] IMPLIES ∃z∈ N.(z=0 OR ∀w∈ N. NOT(w<0))
= **[∃x∈ N. (x=y)] IMPLIES ∃z∈ N.∀w∈ N.(z=0 OR NOT(w<0))** = ∀x∈ N. [(x=y) IMPLIES ∃z∈ N.∀w∈ N.(z=0 OR NOT(w<0))]
= ∀x∈ N. ∃z∈ N. ∀w∈ N. [(x=y) IMPLIES (z=0 OR NOT(w<0))]

Apply an alternative sequence of transformations:
[∃x∈ N. (x=y)] IMPLIES ∃z∈ N.∀w∈ N.(z=0 OR NOT(w<0)) = NOT[∃x∈ N. (x=y)] OR ∃z∈ N.∀w∈ N.(z=0 OR NOT(w<0))
= [∀x∈ N. NOT(x=y)] OR ∃z∈ N.∀w∈ N.(z=0 OR NOT(w<0)) = ∃z∈ N.∀w∈ N. ∀x∈ N. [NOT(x=y) OR (z=0 OR NOT(w<0))]

Online Quiz 1-2

Online Quiz *:
Define the following propositions: P=Visit the doctor, Q=Get a blood test, R=I am nervous
Which of the following is equivalent to (P IMPLIES Q) IMPLIES R?
I get nervous whenever visiting the doctor requires getting a blood test done.

Online Quiz 1:
For the following questions:
Let D=a non-empty set of questions, and suppose that each question in D is either an assignment question or an exam question but not both.
For all x∈D, let E(x)='x is an exam question', L(x)='x is long'
For all x, y∈D, let H(x, y)='x is harder than y'

①Every assignment question is short
∀x∈D, NOT(E(x)) IMPLIES NOT(L(x))
There is no assignment question that is short.
NOT ∃x∈D.((NOT E(x)) AND NOT L(x))
Every short question is an assignment question.
∀x∈D.((NOT L(x)) IMPLIES (NOT E(x)))
There are no short questions and no assignment questions. There are only long exam questions.
NOT ∃x∈D.((NOT L(x)) OR (NOT E(x))) = ∀x∈D. L(x) AND E(x)
②∃x∈D.(E(x) AND L(x))
There is a long exam question. Not all exam questions are short.
③Some assignment question is harder than every exam question
∃x∈D, ∀y∈D, NOT(E(y)) AND [E(x) IMPLIES H(x, y)]
For each assignment question, there is an easier exam question.
$\forall y \in D.\ \exists x \in D.([(\text{NOT } E(y)) \text{ AND } E(x)] \text{ IMPLIES } H(y, x))$
There is an assignment question harder than all exam questions.
$\exists x \in D.\forall y \in D.[(\text{NOT } E(x)) \text{ AND } E(y) \text{ AND } H(x, y)]$
④NOT ∃x∈D.[E(x) AND ∀y∈D.[(NOT E(y)) IMPLIES H(x, y)]]
No exam question is harder than all assignment questions.
Some assignment question is at least as hard as every exam question.
For every exam question, there is an assignment question that is at least as hard.
⑤∀x∈D.([E(x) AND L(x)] IMPLIES [∃y∈D.[(NOT E(y)) AND NOT H(x, y)]])
For every long exam question, there is an assignment question that is at least as hard.

Online Quiz 2:For the following questions:
Let A be a non-empty set of animals, and suppose that every animal in A is either a cat or a dog.
For all x∈A, let B(x)='x is black'
For all x∈A, let C(x)='x is a cat'
For all x∈A, let D(x)='x is a dog'
For all x, y∈A, let T(x, y)='x is taller than y'
Also, suppose that we are given the predicate EQUALS(x, y) which is true when x and y are the same animal, and false otherwise.

①All cats are black	∀x∈A, C(x) IMPLIES B(x)
②Given that ∃x∈A. (D(x) AND NOT B(x)) is true,	It is not the case that all dogs are black.

③the tallest animal is a dog.
∀x∈A.[C(x) IMPLIES ∃y∈A.(D(y) AND T(y, x))]
ALMOST! It depends on how the English statement is interpreted. In many cases, "THE tallest animal" implies that there is a unique tallest animal. This answer says that, for each cat in the domain, there is a taller dog. This would certainly imply that the tallest animal is not a cat. However, this does not mean that the tallest animal is a dog, because there may not necessarily be a tallest animal! For example, this answer will evaluate to true if there are two or more dogs of the same height that are taller than all cats.

∃x∈ A.∀y∈ A.[D(x) AND ((NOT(T(x, y)) IMPLIES EQUALS(x, y))]]

This says that there exists some animal x such that x is a dog, and, if x is not taller than some animal y, then x=y. Unlike previous answer, this implies that there is one animal that is the tallest.

④Given∀x∈ A. [(C(x) AND B(x)) IMPLIES (∀y∈ A. (D(y) AND B(y)) IMPLIES T(y, x))],

No black cat is taller than any black dog.

⑤There is exactly one black dog.

∃x∈ A. (D(x) AND B(x) AND ∀y∈ A. [(D(y) AND B(y)) IMPLIES EQUALS(x, y)])

This answer says that there exists x that is a black dog, and, if y is a black dog, then y=x. Therefore, there is at least one black dog, and less than two black dogs.

∀*x*∈ *A*. ∀*y*∈ *A*. [(*D*(*x*) AND B(x) AND D(y) AND *B*(*y*)) IMPLIES EQUALS(*x*, *y*)] ✘

INCORRECT! This answer says that, if x and y are black dogs, then x=y. This means that there is at most one black dog, which includes the possibility of zero black dogs.

Online Quiz 3-4

Online Quiz 1:

M IMPLIES Q Satisfiable, not valid (True, M=T, Q=T, or M=F, Q=T/F; False, M=T, Q=F)

M IMPLIES (NOT P OR NOT Q)

M IMPLIES [M AND (P IMPLIES M)]

(P OR Q) IMPLIES Q

(PORQ)IMPLIES(NOTPAND NOTQ)

(P OR Q) IMPLIES [M AND (P IMPLIES M)]

(P XOR Q) IMPLIES Q

(PXORQ)IMPLIES(NOTPOR NOTQ)

(P XOR Q) IMPLIES [M AND (P IMPLIES M)] M IMPLIES Q

Online Quiz 4: CNF for (A XOR B) XOR C

A	B	C	A XOR B	(A XOR B) XOR C	Negation	
T	T	T	F	T		
T	T	F	F	F	T	A AND B AND NOT(C)
T	F	T	T	F	T	A AND NOT(B) AND C
T	F	F	T	T		
F	T	T	T	F	T	NOT(A) AND B AND C
F	T	F	T	T		
F	F	T	F	T		
F	F	F	F	F	T	NOT(A) AND NOT(B) AND NOT(C)

NOT{[A AND B AND NOT(C)] OR [A AND NOT(B) AND C] OR [NOT(A) AND B AND C] OR [NOT(A) AND NOT(B) AND NOT(C)]}

= [NOT(A) OR NOT(B) OR C)] AND [NOT(A) OR B OR NOT(C)] AND [A OR NOT(B) OR NOT(C)] AND [A OR B OR C]

Online Quiz 5:

①[∃x∈ D. P(x)] IMPLIES [∀y∈ D. P(y)]

Not valid (Let D=N, for all x∈ D, let P(x)='x is even'. x=2, y=3 makes hypothesis true but conclusion false)

②[(∃x∈ D. P(x)) AND (∃y∈ D. Q(y))] IMPLIES [∃z∈ D. (P(z) AND Q(z))]

Not valid (Let D=Z, for all x∈ D, let P(x)='x is negative' and Q(x)='x is greater than 1'. x=-1, y=2)

③[∃x∈ D.∃y∈ D.P(x, y)] IMPLIES [∃z∈ D.∃w∈ D.P(w, z)]

Valid (Suppose that there exist x, y∈ D such that P(x, y) is true. Then, we can choose z=y and w=x, which makes P(w, z) true. Therefore, whenever the hypothesis of the implication is true, so is the conclusion)

④[∀x∈ D.∃y∈ D.Q(x, y)] IMPLIES [∃z∈ D.∀w∈ D.Q(w, z)]

Not valid (Let D=N, for all x, y∈ D, let Q(x, y)='x≤y')

⑤[∃x∈ D.∀y∈ D.R(x, y)] IMPLIES [∀z∈ D.∃w∈ D.R(w, z)]

Valid (Suppose that there exists an x∈ D such that, for all y∈ D, R(x, y) is true. Then, for all z∈ D, it must be the case that R(x, z) is true. Namely, choosing w=x makes R(w, z) true for all choices for z. Therefore, the conclusion is true.)

⑥(∀x∈ D.∃y∈ D.P(x, y)) IMPLIES (∀z∈ D.P(z, z))

Not valid (Let D=N, for all x, y∈ D, let P(x, y)='x*y is even')

Online Quiz 6:For each of the following formulas, determine whether or not it is true in each of the domains $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$. (Assume that the constant symbols map to their usual values in the chosen domain, e.g the constant symbol 2 maps to the number 2)

①$\forall x\in D.\ \exists y\in D.\ 2x-y=0$ (True for domains N, Z, Q, R, C)

②$\forall x\in D.\ \exists y\in D.\ x-2y=0$ (False for N and Z, True for the rest)

③$\forall x\in D.[(x<10)$ IMPLIES $(\forall y\in D.(y<x)$ IMPLIES $(y<9))]$ (True for N and Z)

④$\forall x\in D.\ \exists y\in D.\ [(y > x)$ AND $(\exists z\in D.\ y+z=100)]$ (False for N, x=100 is a counterexample)

⑤$\exists x\in D.x2 =2$ (True for R, C)

⑥$\forall x\in D.\exists y\in D.x2=y$ (True for all, they are all closed under multiplication)

⑦$\forall y\in D.\exists x\in D.x2=y$ (True for C)

⑧$\forall x\in D.\ \exists y\in D.\ [(x\neq 0)$ IMPLIES $(x\cdot y=1)]$ (True for Q, R, C)

⑨$\exists x\in D.\ \exists y\in D.\ [(x+2y=2)$ AND $(2x+4y=5)]$

(False for all. If it were true, from the first equation, x=2-2y; Substitute into the second, 2(2-2y)+4y=5, which implies 4=5, which is impossible. Proof by contradiction!)

Online Quiz 8: Prenex Normal Form of $(\forall x\in D.\ [\forall z\in C.\ [(\exists x\in D.\ S(z, x))$ IMPLIES $P(x, y, z)]])$ AND $M(z)$
Rename quantified variables that also appear free. In this case, the final occurrence of z is free and the rest are quantified.
$(\forall x\in D.[\forall v\in C.[(\exists u\in D.S(v, u))$ IMPLIES $P(x, y, v)]])$ AND $M(z)$
Move AND M(z) inside the quantifier $\forall v\in C$.
$\forall x\in D.\forall v\in C.([(\exists u\in D.S(v, u))$ IMPLIES $P(x, y, v)]$ AND $M(z))$
Move IMPLIES P(x, y, v) inside the quantifier $\exists u\in D$ and change $\exists u\in D$ to $\forall u\in D$.
$\forall x\in D.\forall v\in C.([\forall u\in D.(S(v, u)$ IMPLIES $P(x, y, v))]$ AND $M(z))$
Move AND M(z) inside the quantifier $\forall u\in D$.
$\forall x\in D.\forall v\in C.\forall u\in D.([S(v, u)$ IMPLIES $P(x, y, v)]$ AND $M(z))$

Tutorial 3 or 4

Prove $A\subseteq B$
$\forall x\in A, x\in B$
$\forall x\in U, x\in A$ IMPLIES $x\in B$
1.Let $x\in A$ be arbitrary.
…
n-2. $x\in B$
n-1. $x\in A$ $x\in B$direct proof 1, n-2
n. $\forall x\in A, x\in B$

Prove A=B $A\subseteq B, B\subseteq A$
Prove $|A|=|B|$ $f: A\rightarrow B$ bijective
bijective $\forall b\in B$, there exists exactly one a in A such that f(a)=b.

E.g.Let $A=\{a0,\dots a_{m-1}\}$, $B=\{b0,\dots b_{n-1}\}$ $f: |A\times B|\rightarrow\{0, 1, \dots mn-1\}$ $|A\times B|=mn$

Ramsey's theorem: R(3, 3)=6
If each edge of the complete graph on 6 vertices is colored with one of 2 colors, then there is one monochromatic triangle.
Let x be vertex, N(x)='neighbours of x' and N(x)={y1, y2, y3, y4, y5},
{x, y1}, {x, y2}, {x, y3} are blue
If {y1, y2} is blue, then Δxy1y2 blue
If {y2, y3} is blue, then Δxy2y3 blue
If {y3, y1} is blue, then Δxy3y1 blue
Δy1y2y3 red

Problem Session 5

Problem 5.1: ((A IMPLIES B) AND (NOT(A) IMPLIES B)) IMPLIES B

1.((A IMPLIES B) AND (NOT(A) IMPLIES B))	assumption
2.Assume NOT(B)	
3.A IMPLIES B	use of conjunction 1
4.NOT(B) IMPLIES NOT(A)	contrapositive 3
5.NOT(A)	modus ponens 2,4
6.NOT(A) IMPLIES B	use of conjunction 1
7.NOT(B) IMPLIES A	contrapositive 6
8.A	modus ponens 2,7
9.B	Proof by contradiction 2,5,8
10. ((A IMPLIES B) AND (NOT(A) IMPLIES B)) IMPLIES B	direct proof 1,9

Problem 5.2: Suppose that R is logically equivalent to S, R' is obtained from R by making a substitution, S' is obtained from S by making the same substitution, and R' is true. Prove S' is true or give a counterexample.

①Let R=(P IMPLIES Q) IMPLIES P, S= P AND P, R'=(P IMPLIES Q) IMPLIES U, S'=P AND U.
R is logically equivalent to S (check their truth tables), R' is obtained from R by substituting U for the second P, S' is obtained from S by substituting U for the second P. And, when P=T, Q=U=F, R'=T but S'=F.
(Not all occurrences of P in R are replaced by U)

②Let R=(P OR Q) IMPLIES (P AND Q), S=P IFF (Q OR (P AND Q)), R'=(P OR Q) IMPLIES U, S'=P IFF (Q OR U)
R is logically equivalent to S (check their truth tables), R' is obtained from R by substituting U for (P AND Q), S' is obtained from S by substituting U for (P AND Q). And, when P=Q=F, U=T, R'=T but S'=F.
(Substituting for a formula, rather than a variable)

Suppose that:
R is logically equivalent to S,
R′ is obtained from R by making a substitution (by substituting all occurrences of some propositional variable P by a formula Q), S′ is obtained from S by making the same substitution, and R′ is true.

Proof of S' is true:
1.R' assumption
2.R IFF S assumption
3.R IMPLIES S use of equivalence 2
4.R' IMPLIES S' substitution 3
5.S' modus ponens 1,4

Problem 5.3: Why is the following proof of 1/8 > 1/4 bogus?
$3>2$
$3 \log_{10}(1/2) > 2 \log_{10}(1/2)$
$\log_{10}(1/2)^3 > \log_{10}(1/2)^2$
$1/8 = (1/2)^3 > (1/2)^2 = 1/4$
$\log_{10}(x)<0$ for $0<x<1$, since the second line is obtained from the first by multiplying both sides by a negative number, the inequality needs to be reversed.

Problem 5.4: Why is the following proof bogus? It claims to prove that if a and b are equal real numbers, then a = 0.
$a=b$
$a^2 = ab$
$a^2-b^2 = ab-b2$
$(a-b)(a+b) = (a-b)b$
$a+b=b$
$a=0$
Since a-b=0, you cannot divide both sides of the equation on line 4 to get the equation on line 5.

Problem Session 6

Problem 6.1: give a formal proof with a top down approach

[∀x∈ D.(R(x) IFF ∀y∈ D.A(x, y))] IMPLIES [∃x∈ D.∀y∈ D.(R(y) IMPLIES A(x, y))].

1.Assume ∀x∈ D.(R(x) IFF ∀y∈ D.A(x, y))
2.P OR NOT(P) tautology
3.(∃x∈ D.R(x)) OR NOT(∃x∈ D.R(x)) substitution 2
4. Suppose ∃x∈ D.R(x)
5. Let d∈ D be such that R(d) instantiation 4
6. R(d) IFF ∀y∈ D.A(d, y) specialization 1
7. ∀y∈ D.A(d, y) modus ponens 6,5
8. Let y∈ D be arbitrary.
9. Assume R(y).
10. A(d, y) specialization 7
11. R(y) IMPLIES A(d, y) direct proof 9,10
12. ∀y∈ D.(R(y) IMPLIES A(d, y)) generalization 8,11
13. ∃x∈ D.∀y∈ D.(R(y) IMPLIES A(x, y)) construction 5,12
14. [∃x∈ D.R(x)] IMPLIES [∃x∈ D.∀y∈ D.(R(y) IMPLIES A(x, y))] direct proof 4,13
15. Suppose NOT(∃x∈ D.R(x)).
16. ∀x∈ D.NOT(R(x)) negation of quantifiers 15
17. Let x∈ D be arbitrary.
18. Let y∈ D be arbitrary.
19. Assume NOT(A(x, y)).
20. NOT(R(y)) specialization 16
21. R(y) IMPLIES A(x, y) indirect proof 19, 20
22. ∀y∈ D.(R(y) IMPLIES A(x, y)) generalization 18,21
23. ∃x∈ D.∀y∈ D.(R(y) IMPLIES A(x, y)) construction 17,22
24. [NOT(∃x∈ D.R(x))] IMPLIES ∃x∈ D.∀y∈ D.(R(y) IMPLIES A(x, y))] direct proof 15,23
25. ∃x∈ D.∀y∈ D.(R(y) IMPLIES A(x, y)) proof by cases 3,14,24.
26.[∀x∈ D.(R(x) IFF ∀y∈ D.A(x, y))] IMPLIES [∃x∈ D.∀y∈ D.(R(y) IMPLIES A(x, y))] direct proof 1,25

Problem 6.2: Give a well structured informal proof: Let X be a set and let A,B,C ⊆ X. Suppose that A∩B=A∩C and (X−A)∩B=(X−A)∩C. Then B=C.

You may use use the following lemma without proof: For all sets U andV, U∩V ⊆U and U∩V ⊆V.

Assume A,B,C⊆X, A∩B=A∩C and (X−A)∩B=(X−A)∩C
Let b∈ B.
Since B⊆X, it follows that b∈ X.
b∈ A or b∈ X-A.
Suppose b∈ A
Then b∈ A∩B
Since A∩B=A∩C
it follows that b∈ A∩C
By the lemma, A∩C⊆C
Thus, b∈ C
Therefore b∈ A IMPLIES b∈ C
Suppose b∈ X-A
Then b∈ (X-A)∩B
Since (X-A)∩B=(X-A)∩C
it follows that b∈ (X-A)∩C
By the lemma, (X-A)∩C⊆C
Thus, b∈ C
Therefore b∈ (X-A) IMPLIES b∈ C
Then b∈ C proof by cases
Hence B⊆C.
By symmetry, C⊆B
Thus B=C.

We can use symmetry, because the assumptions and statement do not change when all occurrences of B and C are interchanged.

Online Quiz 5-6

Online Quiz 1: There exists an integer x such that for every integer y, x+y=y.
Proof:
1.Let x=0. Note that x∈Z.
2.Let y be an arbitrary integer.
3.Then x+y=0+y=y (by axioms of arithmetic)
4.(∀y∈Z. x+y=y) is true (by generalization 2,3)
5.(∃x∈Z.∀y∈Z. x+y=y) is true (by construction 1,4)

Online Quiz 2: For every integer y, there exists an integer x such that x+y=0
1.Let y be an arbitrary integer.
2.Let x=-y, Note that x∈Z since y∈Z
3.Then x+y=-y+y=0 (by axioms of arithmetic)
4.(∃x∈Z. x+y=0) is true (by construction 2,3)
5.(∀y∈Z.∃x∈Z. x+y=0) is true (by generalization 1,4)

From the two questions above, notice that the order of the quantifiers changes the nature of the proofs.
To prove a proposition of the form ∀y∈D.∃x∈C.p(x,y) (as in question 2), we pick an arbitrary element y in the domain D and we can use y to construct an x in C.
However, when proving a proposition of the form ∃x∈C.∀y∈D.p(x,y) (as in question 1), we must define x before we pick y, so we cannot use y to construct x.

Online Quiz 3: Suppose that we have a statement of the form (∀x∈D.F). Suppose that we replace each occurrence of the variable x in formula F with the variable y to get a formula F′. Are (∀x∈D.F) and (∀y∈D.F′) equivalent?
It depends whether or not the variable y occurs in F. If y does not occur in F, then (∀x∈D.F) and (∀y∈D.F′) equivalent.
Consider the example: Let F=(A AND NOT(y)). Then F′=(y AND (NOT(y))). Therefore, F′ always evaluates to false.
So under any interpretation that sets (∀x∈D.F) to true, the statements (∀x∈D.F) and (∀y∈D.F′) are not equivalent.

Online Quiz 4: Every non-zero rational number x has a multiplicative inverse. (a rational number y such that xy=yx=1). Prove that this inverse is unique.
To obtain a contradiction, assume that there exist two distinct inverses for x, say a and b. Then, by definition of inverse, we know that ax=1. Multiply both sides by b to get axb=b. Again, by the definition of inverse, xb=1. Substituting this into axb=b, it follows that a=b, which contradicts the fact that a and b are distinct.
Proof: (formal)
1.Consider any x∈Q
2. Assume x≠0
3. Assume ∃a∈Q.∃b∈Q.[(a≠b) AND (ax=1) AND (bx=1)]
4. Choose a,b∈Q such that a≠b and ax=1 and bx=1 (instantiation 3)
5.ax=1 IMPLIES axb=b (axiom of algebra)
6.axb=b (modus ponens 4,5)
7.bx=1 IMPLIES xb=1 (axiom of algebra)
8.xb=1 (modus ponens 4,7)
9.axb=b AND xb=1 (proof of conjunction 6,8)
10.(axb=b AND xb=1) IMPLIES a=b (**tautology**)
11.a=b (modus ponens 9,10)
12.NOT(∃a∈Q.∃b∈Q.[(a≠b) AND (ax=1) AND (bx=1)]) (**contradiction** 3,4,11)
13.x≠0 IMPLIES NOT(∃a∈Q.∃b∈Q.[(a≠b) AND (ax=1) AND (bx=1)]) (direct proof 2,12)
14.∀x∈Q.x≠0 IMPLIES NOT(∃a∈Q.∃b∈Q.[(a≠b) AND (ax=1) AND (bx=1)]) (**generalization** 1,13)

CSC165 Mathematical Expressions and Reasoning

Why does CS need Mathematical Expressions and Reasoning?
—Computer graphics use multi-variable calculus, projective geometry, linear algebra, physics-based modelling
—Numerical analysis uses multivariable calculus and linear algebra
—Cryptography uses number theory, eld theory
—Networking uses graph theory, statistics
—Algorithms use combinatorics, probability, set theory
—Databases use set theory, logic
—AI uses set theory, probability, logic Programming languages use set theory, logic

Mathematical Prerequisites

✳ Set Theory and Notation ✳

A set is a collection of 0 or more (distinct) elements and are often presented as a list surrounded by curly brackets (braces), with a comma between each element. (finite number of elements or infinitely many elements)
Z: integers, or whole numbers {.., -2,-1, 0, 1, 2, ..}.
N: natural numbers or non-negative integers {0, 1, 2, ..}.
The convention in computer science is to include 0 in the natural numbers.
Z^+: positive integers {1, 2, 3, ..} Z^-: negative integers {-1, -2, -3, ..}
Z*: non-zero integers {.., -2,-1, 1, 2, ..} = $Z-\{0\}=Z^-\cup Z^+$.
Q: rational numbers (ratios of integers), comprised of {0}, Q^+ (positive rationals) and Q^- (negative rationals).
The set of all numbers of the form p/q, where $p\in Z$ and $q\in Z^*$.
R: real numbers, comprised of {0}, R^+ (positive reals) and R^- (negative reals).
The set of all numbers of the form m.d1d2d3 , where $m\in Z$ and d1, d2, d3$\in$ {0, 1, 2 ,3, 4, 5, 6, 7, 8, 9}.

Operations on Sets, For any sets A and B:
$x\in A$: x is an element of A, x is in A, A contains x.
$A\subseteq B$: A is a subset of B, Every element of A is also an element of B.(A includes only elements of B) A is included in B.
A set is always a subset of itself.
A=B: A equals B, A and B contain exactly the same elements, A=B IFF [$A\subseteq B$ and $B\subseteq A$].
$A\cup B$: A union B, the set of elements that are in either A or B, or both.
$A\cap B$: A intersection B, the set of elements that are in both A and B.
A\B or A−B: A minus B, the set of elements that are in A but not in B (set difference).
A×B: (Cartesian) product of *A* and *B*, the set consisting of all *pairs* (*a*, *b*) where *a* is an element of *A* and *b* is an element of *B*.
A_bar: complement of A
|A|: cardinality of A, the number of elements in A, the size of the finite set A.
∅ or {}: empty set, a set that contains no elements. Empty set is a subset of any set.
By convention, for any set A, $\emptyset\subseteq A$ (we will see a logical justification for this fact when we discuss vacuous truth).
P(A): power set of A, the set of all subsets of A. e.g, suppose A={73, a}, then P(A)={∅, {73}, {a}, {73, a}}.
Proper Subset: $A\subsetneq B$, ($A\subsetneq B$ IFF $A\subseteq B$ and $A\neq B$)
{x: P(x)} or {x| P(x)}: The set of all x for which P(x) is true. e.g, $\{x\in Z: \cos(\pi x)>0\}$={.., -2, 0, 2, ..} (even integers).
Set Builder {x∈ A| P(x)}: the set of elements x in A for which P(x) is true
e.g. $\{x\in Z \mid \cos(\pi x)>0\}$ represents the set of integers x for which cos(πx) is greater than zero,
i.e {..-4, -2, 0, 2, 4..}={x∈Z| x is even}

✳ Number theory ✳

If m and n are natural numbers, with $n\neq 0$, then there is exactly one pair of natural numbers (q, r) such that:
$m=q*n+r$, $n>r\geq 0$ q is the quotient of m divided by n, and r is the remainder m mod n = r
When the remainder r is zero(m = q*n), n divides m and write n | m, n is a divisor of m (e.g. 4 is a divisor of 12).
For any two natural numbers a and b, a divides b if there exists a natural number c such that b=ac. a is a divisor of b(e.g 3 is a divisor of 12 but 3 is not a divisor of 16).
Note: Any natural number is a divisor of 0, and that 1 is a divisor of any natural number.
A natural number p is prime if it has exactly two positive divisors. (2, 3, 5, 7… are all prime) but 1 is not (too few positive divisors, it only has one positive divisor, 1) and 9 is not (too many positive divisors).

There are an infinite number of prime numbers and any integer greater than 1 can be expressed in a unique way as a finite product of prime numbers (a product of one or more primes). e.g $8 = 2^3$, $77 = 7 \times 11$, $3 = 3$

Let $n, d \in Z$, d **divides** n, n **is divisible by** d, or $d \mid n$, when there exists a $k \in Z$ such that $n = dk$, d a divisor of n, and n a multiple of d.

Quotient-Remainder Theorem: For all $n \in Z$ and $d \in Z+$, there exist $q, r \in Z$ such that $n=qd+r$ and $0 \le r < d$. Moreover, these q and r are unique (they are determined entirely by the values of n and d), q and r are the quotient and remainder, respectively, when n is divided by d.

✳ Functions ✳

(x, y are arbitrary real numbers, k, m and n are arbitrary positive integers)

f is a function/mapping from set A to set B, $f : A \to B$, A is domain of the function, B is range of the function.

For every $x \in A$ there is an associated $f(x) \in B$. (f associates at most one element $f(x) \in B$)

Functions can have more than one input. For sets $A_1, A_2, \ldots, A_k$ and B, a ***k*-ary function** $f : A_1 \times A_2 \times \cdots \times A_k \to B$ is a function that takes k arguments, where for each i between 1 and k, the i-th argument of f must be an element of A_i, and where f returns an element of B. (unary function takes one input, binary two, ternary three)

Predicate: a function whose range is {True, False}

Note that variables $x, y \in R$ whereas $m, n \in Z+$.

$\min\{x, y\}$: minimum of x or y, the smaller of x or y. $\min\{x, y\} \le x$ and $\min\{x, y\} \le y$

$\max\{x, y\}$: maximum of x or y, the larger of x or y. $x \le \max\{x, y\}$ and $y \le \max\{x, y\}$

$|x|$: absolute value of x, which is x if $x \ge 0$, is -x if $x < 0$

gcd(m, n): greatest common divisor of m and n, the largest positive integer that divides both m and n.

lcm(m, n): least common multiple of m and n, the smallest positive integer that is a multiple of both m and n.

$\gcd(m, n) * \mathrm{lcm}(m, n) = m \cdot n$

$\lfloor x \rfloor$ or floor(x): largest integer that is not larger than x, $\forall x \in R, y = \lfloor x \rfloor \Leftrightarrow y \in Z \wedge y \le x \wedge (\forall z \in Z, z \le x \Rightarrow z \le y)$

$x - 1 < \lfloor x \rfloor \le x$, $\lfloor -x \rfloor = -\lceil x \rceil$, $\lfloor x+k \rfloor = \lfloor x \rfloor + k$, $\lfloor \lfloor k/m \rfloor / n \rfloor = \lfloor k/mn \rfloor$, $(k - m + 1)/m \le \lfloor k/m \rfloor$

$\lceil x \rceil$ or ceil(x): smallest integer that is not smaller than x, $\forall x \in R, y = \lceil x \rceil \Leftrightarrow y \in Z \wedge y \ge x \wedge (\forall z \in Z, z \ge x \Rightarrow z \ge y)$

$x \le \lceil x \rceil < x + 1$, $\lceil -x \rceil = -\lfloor x \rfloor$, $\lceil x+k \rceil = \lceil x \rceil + k$, $\lceil \lceil k/m \rceil / n \rceil = \lceil k/mn \rceil$, $\lceil k/m \rceil \le (k + m - 1)/m$.

$\lfloor k/2 \rfloor + \lceil k/2 \rceil = k$

m div n: quotient of m divided by n, integer division of m by n (e.g 5 div 6 = 0, 27 div 4 = 6, –27 div 4 = –6)

If $m, n > 0$, then $m \text{ div } n = \lfloor m/n \rfloor$ $(-m) \text{ div } n = -(m \text{ div } n) = m \text{ div } (-n)$.

m rem n: remainder of m divided by n (e.g 5rem6 = 5, 27rem4 = 3, –27rem4 = –3)

$m = (m \text{ div } n) \cdot n + m \text{ rem } n$ $(-m) \text{ rem } n = -(m \text{ rem } n) = m \text{ rem } (-n)$.

m mod n: m modulo n (e.g 5 mod 6=5, 27 mod 4=3, –27 mod 4=1)

$0 \le m \bmod n < n$, n divides $m - (m \bmod n)$

Summation Notation: for any pairs of integers j and k, and any function f: Z->R

$$\sum_{i=j}^{k} f(i) = f(j) + f(j+1) + \ldots + f(k)$$ i: index of summation, j & k: lower & upper bounds of summation

Product Notation: $$\prod_{i=j}^{k} f(i) = f(j) \times f(j+1) \times \ldots \times f(k)$$

A summation/product's lower bound to be greater than its upper bound, the summation/product is empty.

$\Sigma_{i=j}^{k} f(i) = 0$, $\prod_{i=j}^{k} f(i) = 1$.

✳ Inequalities ✳

For any $m, n \in Z$ (integers): $m < n$ if and only if $m+1 \le n$, and $m > n$ if and only if $m \ge n+1$.

For any $x, y, z, w \in R$ (real numbers):

If $x < y$ and $w \le z$, then $x + w < y + z$.

If $x < y$, then $xz < yz$ if $z > 0$, $xz = yz$ if $z = 0$, $xz > yz$ if $z < 0$.

If $x < y$ and $y \le z$(or $x \le y$ and $y < z$),then $x < z$.

If $x \le y$, then $x+z \le y+z$.

If $0 < a \le b$, then $1/a \ge 1/b$. If $a \le b < 0$, then $1/a \le 1/b$.

triangle inequality $|x+y| \le |x| + |y|$

✳ Calculus ✳

A sequence of real numbers $\{a_n\}=a0, a1, a2,.., a_n,...$ converges to a limit $L\in R$ if for every $\varepsilon>0$, there exists an $n0\geq 0$ such that $|an-L|<\varepsilon$ for every $n\geq n0$. In such a case, we write $\lim_{n\to\infty}a_n=L$ or simply $\{a_n\}\to L$. Otherwise, we say that the sequence diverges.

If $\{a_n\}$ and $\{b_n\}$ are two sequences of real numbers such that $\{a_n\}\to L_1$ and $\{b_n\}\to L_2$, then $\lim_{n\to\infty}(a_n+b_n)=L_1+L_2$ and $\lim_{n\to\infty}(a_n\cdot b_n) = L_1\cdot L_2$. In particular, if c is any real number, then $\lim_{n\to\infty}(c\cdot a_n)=c\cdot L_1$.

For any $a\in R$ such that $-1<a<1$, $\lim_{n\to\infty}a^n=0$.

For any $a\in R+$, $\lim_{n\to\infty}a^{1/n}=1$.

For any $a\in R+$, $\lim_{n\to\infty}(1/n)^a=0$.

$\lim_{n\to\infty}(1+1/n)^n=e=2.71828182845904523536\ldots$

For any a, $b\in R$, arithmetic sum is given by: $\sum_{i=0}^{n}(a+ib)=(a)+(a+b)+(a+2b)+..+(a+nb)=\frac{1}{2}(n+1)(2a+nb)$

For any a, $b\in R^+$, geometric sum is given by: $\sum_{i=0}^{n}(ab^i)=a+ab+ab^2..+ab^n=\frac{a(1-b^{n+1})}{1-b}$

✳ Exponents & Logarithms ✳

For any a, b, $c\in R^+$: $a=\log_b c$ if and only if $b^a=c$. For any $x\in R^+$: $\ln x=\log_e x$ and $\lg x=\log_2 x$

For any a, b, $c\in R^+$ and $n\in Z^+$:

- $\sqrt[n]{b}=b^{1/n}$
- $b^a b^c=b^{a+c}$
- $(b^a)^c=b^{ac}$
- $b^a/b^c=b^{a-c}$
- $b^0=1$
- $a^b c^b=(ac)^b$
- $b^{\log_b a}=a=\log_b b^a$
- $a^{\log_b c}=c^{\log_b a}$
- $\log_b(ac)=\log_b a+\log_b c$
- $\log_b(a^c)=c\cdot\log_b a$
- $\log_b(a/c)=\log_b a-\log_b c$
- $\log_b 1=0$
- $\log_b a=\log_c a/\log_c b$

✳ Binary Notation ✳

A binary number is a sequence of bits ak…a1a0, where each bit ai is equal to 0 or 1. Every binary number represents a natural number: $(a_k..a_1a_0)_2=\sum_{i=0}^{k}a_i 2^i=a_k 2^k+..+a_1 2+a_0$

e.g $(1001)_2=1\cdot 2^3+0\cdot 2^2+0\cdot 2^1+1\cdot 2^0=9$ $(01110)_2=8+4+2=14$

If $a=(a_k\cdots a_1a_0)_2$, then $2a=(a_k\cdots a_1a_00)_2$, e.g 9 = (1001)2 so 18=(10010)2.

If $a=(a_k\cdots a_1a_0)_2$, then $\lfloor a/2\rfloor=(a_k\cdots a_1)_2$, e.g 9 = (1001)2 so 4=(100)2.

The smallest number of bits required to represent the positive integer n in binary is called the length of n and is equal to $\lceil\lg(n+1)\rceil$.

Logical Notation

Universal Quantification

∀(for all)

When a claim is made about all the objects(every member of a class/universe/domain) being considered.

Every employee makes less than 70000. = Each Employee makes less than 70000. = All employees make less than 70000. = Employees make less than 70000

To disprove/refute a universally-quantified statement, one counterexample is sufficient.

To prove a universally-quantified statement, verify every element in a domain.

Existential Quantification

✳∃(there exists)

Claims are about the existence of one or more elements of a domain with some property

Some employee earns over 57000. There is an employee who earn less than 57000.

There is/exists a/an/some/at least one .. such that/for which .. .

For a/an/some/at least one .. , .. .

Note: some is always used inclusively, “some object is a P” is true if every project is a P.

To prove a existential quantification, need to exhibit just one example of an element with the property.

To disprove, need to consider the entire domain to show that every element is a counter-example.

The anti-symmetry between universal and existential quantification may be better understood by switching point of view from properties to the sets of elements having those properties.

✳ **Properties, Sets, Quantification** ✳
a little confusing

Sentences, Statements, Predicates

The employee makes less than 55000. (about a particular employee, true/false depends on the earnings of that employee)
Every employee makes less than 55000. (about the entire set of employees E, true/false depends on where that set of employees stands in relation to the set L, those who earn over 55000)
Sentence: may refer to unquantified objects; once the objects are specified, a sentence is either true or false, but never both. (claim 1) Open sentence: a sentence refers to unquantified objects
Statement: does not refer to any unquantified variables, either true or false. (claim 2)
Every statement is a sentence, but not every sentence is a statement.
A sentence is a statement if and only if it is not open.
Predicate: a boolean function

Implication P⇒Q (if P, then Q)

P: antecedent/assumption Q: consequent/conclusion
To disprove implication "if P then Q", show an instance where P is true but Q is false.
If in every possible instance, either not-P or Q (either P is false or Q is true), then implication "if P then Q" is true.

Converse of P⇒Q is Q⇒P Contrapositive of P⇒Q is ¬Q⇒¬P

Everyday English: P implies Q
If/When(ever) P, (then) Q.
P is sufficient/enough for Q. (It is sufficient that P for Q)
Can't have P without Q.
For P to be true, Q must be true/needs to be true/ is necessary. (It is necessary that Q for P)
P requires Q.
Q if P.
P only if/when Q.
Note: For the antecedent (P) look for "if", "when", "enough", "sufficient".
For the consequent (Q) look for "then","requires","must","need", "necessary", "only if", "when".
In all cases, check whether the expected meaning in English matches the meaning of P⇒Q.
In other words, you've got an implication if, in every possible instance, either P is false or Q is true.

Unless/Despite/Not Withstanding (Ambiguity!!!):
(Not P) Unless Q = Not Q Implies Not P = P Implies Q or sometimes P iff Q

If P, then Q OR R. = If P AND (NOT Q), then R.

Vacuous Truth: whenever the antecedent is false and the consequent is either true or false, the implication as a whole is true. (the set where the antecedent is true is empty(vacuous), and hence a subset of every set)

Equivalence P⇔Q (P iff Q, P if and only if Q) Bi-conditional, Bi-implication

P is necessary and sufficient for Q
P⇒Q and conversely
P (exactly/precisely) when Q. (x2+4x+4=0 precisely when x=-2)
x2-2x+2=0⇔x>x+5 (equivalence since implications are vacuously true in both directions)

Restricting Domains

implication, quantification, conjunction(and, Λ) and set intersection are techniques that can be used to restrict domains
Every D that is also a P is also a Q. ∀x∈D, P(x)⇒Q(x) or ∀x∈D∩P, Q(x)

Why not $\forall x \in D, P(x) \wedge Q(x)$??
My-note: Every x in D is P and Q. (in fact not,)
Some D that is also a P is also a Q. $\exists x \in D, P(x) \wedge Q(x)$ or $\exists x \in D \cap P, Q(x)$
Why not $\exists x \in D, P(x) \Rightarrow Q(x)$??
My-note: There exists a x in D, if P(x) then Q(x).
(If P(x) is false, then $\exists x \in D, P(x) \Rightarrow Q(x)$ is always true, while there does not exist one x in D is P)

Conjunction (AND, ∧)

combine two sentences into a new sentence that claims both of the original sentences are true
$P \wedge Q$ is true exactly when both P and Q are true, and false if only one of them is true and the other is false, or both are false.

conjunction(∧) v.s intersection(∩):

Disjunction (OR, ∨)

join two sentences into one that claims that at least one of the sentences is true (inclusive-or)
exclusive-or: one or the other, but not both
disjunction(∨) v.s union(∪)

Negation (¬P)

negation of a sentence inverts its truth value;
Negation of a sentence P has the value true if P was false, and has the value false if P was true.

"No P is a Q" is equivalent to "Every P is a non-Q"
$\neg(\exists x \in D, P(x) \wedge Q(x)) \Leftrightarrow \forall x \in D, (P(x) \Rightarrow \neg Q(x))$
"Not every P is a Q" is equivalent to "There is some P that is a non-Q"
$\neg(\forall x \in D, P(x) \Rightarrow Q(x)) \Leftrightarrow \exists x \in D, (P(x) \wedge \neg Q(x))$

Negation of "$\forall x \in D, \exists y \in D, P(x, y)$"

$\neg(\forall x \in D, \exists y \in D, P(x, y))$	there is some x for which the remainder of the sentence is false
$\Leftrightarrow \exists x \in D, \neg(\exists y \in D, P(x, y))$	there are no y's for which the remainder of the sentence is true
$\Leftrightarrow \exists x \in D, \forall y \in D, \neg P(x, y)$	there is some x that for all y makes P(x, y) false

Negation of "$\exists x \in D, \forall y \in D, P(x, y)$"

$\neg(\exists x \in D, \forall y \in D, P(x, y))$	there is no x such that the remainder of the sentence is true
$\Leftrightarrow \forall x \in D, \neg(\forall y \in D, P(x, y))$	for all x the remainder of the sentence is false
$\Leftrightarrow \forall x \in D, \exists y \in D, \neg P(x, y)$	for every x there is some y that makes P(x, y) false

Symbolic Grammar:
with connectives (implication, conjunction, disjunction) added to quantifiers, we can form complex predicates.
If you require complex predicates to be unambiguous, it helps to impose strict conditions on what expressions are allowed.
A syntactically correct sentence is called a well-formed formula.
Syntactic correctness has nothing to do with whether a sentence is true or false, or whether a sentence is open or closed.

Syntax(Grammar rules) for our symbolic language:
Any predicate is a wff.
If P is a wff, so is $\neg P$.
If P and Q are wffs, so is $(P \wedge Q)$.
If P and Q are wffs, so is $(P \vee Q)$.
If P and Q are wffs, so is $(P \Rightarrow Q)$.
If P and Q are wffs, so is $(P \Leftrightarrow Q)$.
If P is a wff (possibly open in variable x) and D is a set, then $(\forall x \in D, P)$ is a wff.

If P is a wff (possibly open in variable x) and D is a set, then (∃x∈D, P) is a wff.
Nothing else is a wff.
There rules are recursive, and tell us how we are allowed to build arbitrarily complex sentences in our symbolic language. The first rule is called the base case and specifies the most basic sentence allowed. The rules following the base case are recursive or inductive rules: they tell us how to create a new legal sentence from smaller legal sentences. The last rule is a closure rule, and says we have covered everything.
In practice, use precedence to disambiguate expressions that are missing parentheses. (precedence decreases from top to bottom; in the absence of parentheses, parentheses must be added to sub-expressions near the top before those near the bottom)
e.g. ∀x∈D, P(x)∧¬Q(x)⇒R(x) (∀x∈D, ((P(x)∧¬Q(x)) ⇒ R(x)))

Truth Table

Predicates evaluate to either true or false once they are completely specified (all unknown values are filled in)
In a truth table, write all possible truth values for the predicates, and compute the truth value of the statement under each of these truth assignments.

P	¬P	P∧¬P	P∨¬P	Q	P∧Q	P∨Q	P⇒Q	P⇔Q
F	T	F	T	F	F	F	T	T (!!)
F				T	F	T	T	F
T	F	F	T	F	F	T	F	F
T				T	T	T	T	T

Break complex statements into simpler sub-statements, compute the truth value of the sub-statements and combine the truth values back into the more complex statements.
[Example]: verify the equivalence P⇒(Q⇒R) ⇔ ((P∧Q)⇒R)

Tautology, Satisfiability, Unsatisfiability

Tautology: with truth tables we explored all possible worlds (configurations of truth assignments to P and Q) you can't dream up a domain or a meaning for predicates P and Q that provides a counter-example, since the truth tables are identical.

Satisfiability: statement is true for some choice of domain, there are also choices of domains and/or predicates which it is false.
Note: saying that a statement is satisfiable only tells us that it is possible for it to be true, without saying anything about whether or not it is also possible for it to be false (whether or not it is also a tautology)
Unsatisfiable/Contradiction: no domains, predicates or values can be chosen to make it true

Logical Arithmetic

①**Commutative**: P∧Q ⇔ Q∧P P∨Q ⇔ Q∨P $P \wedge Q \Leftrightarrow Q \wedge P \quad P \vee Q \Leftrightarrow Q \vee P \quad (P \Leftrightarrow Q) \Leftrightarrow (Q \Leftrightarrow P)$

②**Associative**: P∧(Q∧R) ⇔ (P∧Q)∧R P∨(Q∨R) ⇔ (P∨Q)∨R

$(P \wedge Q) \wedge R \Leftrightarrow P \wedge (Q \wedge R) \quad (P \vee Q) \vee R \Leftrightarrow P \vee (Q \vee R)$

③**Distributive**: P∧(Q∨R) ⇔ (P∧Q)∨(P∧R) P∨(Q∧R) ⇔ (P∨Q)∧(P∨R)
P AND (Q OR R) ⇔ (P AND Q) OR (P AND R)
P OR (Q AND R) ⇔ (P OR Q) AND (P OR R)
④**Identity**: P∧(Q∨¬Q) ⇔ P ⇔ P∨(Q∧¬Q) $P \wedge (Q \vee \neg Q) \Leftrightarrow P \Leftrightarrow P \vee (Q \wedge \neg Q)$
P AND (Q OR ¬Q) ⇔ P ⇔ P OR (Q AND ¬Q)
⑤**Idempotency**: P∧P ⇔ P ⇔ P∨P $P \wedge P \Leftrightarrow P \Leftrightarrow P \vee P$
P AND P ⇔ P ⇔ P OR P

deMorgan 's Law: (can be verified by a truth table or by representing the sentences as Venn diagrams and taking the complement)
Sentence s1∧s2 is false exactly when at least one of s1 or s2 is false. ¬(s1∧s2) ⇔ (¬s1∨¬s2) $\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$

Sentence s1∨s2 is false exactly when both s1 and s2 are false. ¬(s1∨s2) ⇔ (¬s1∧¬s2) $\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$

Double Negation: P ⇔ ¬¬P

Implication: (P⇒Q) ⇔ (¬P∨Q) $P \Rightarrow Q \Leftrightarrow \neg P \vee Q$

Implication Negation: ¬(P⇒Q) ⇔ ¬(¬P∨Q) ⇔ (P∧¬Q)

bi-implication: (P⇔Q) ⇔ ((P∧Q)∨(¬P∧¬Q)) ¬(P⇔Q) ⇔ ¬((P∧Q)∨(¬P∧¬Q)) …

Contrapositive: (P⇒Q) ⇔ (¬Q⇒¬P) $P \Rightarrow Q \Leftrightarrow \neg Q \Rightarrow \neg P$

Equivalence: $(P \Leftrightarrow Q) \Leftrightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow P)$

Equivalence Negation: $\neg(P \Leftrightarrow Q) \Leftrightarrow \neg(P \Rightarrow Q) \vee \neg(Q \Rightarrow P)$

Quantifier Negation: $\neg(\forall x \in D, P(x)) \Leftrightarrow \exists x \in D, \neg P(x)$ $\neg(\exists x \in D, P(x)) \Leftrightarrow \forall x \in D, \neg P(x)$

$$\begin{aligned}
\forall x \in D, P(x) \wedge Q(x) &\iff (\forall x \in D, P(x)) \wedge (\forall x \in D, Q(x)) \\
\exists x \in D, P(x) \vee Q(x) &\iff (\exists x \in D, P(x)) \vee (\exists x \in D, Q(x)) \\
\forall x \in D, R \wedge Q(x) &\iff R \wedge (\forall x \in D, Q(x)) \\
\forall x \in D, R \vee Q(x) &\iff R \vee (\forall x \in D, Q(x)) \\
\exists x \in D, R \vee Q(x) &\iff R \vee (\exists x \in D, Q(x)) \\
\exists x \in D, R \wedge Q(x) &\iff R \wedge (\exists x \in D, Q(x))
\end{aligned}$$

Quantifier Distributive Laws: where R does not contain variable x

Variable Renaming: where y does not appear in P(x) $\forall x \in D, P(x) \Leftrightarrow \forall y \in D, P(y)$ $\exists x \in D, P(x) \Leftrightarrow \exists y \in D, P(y)$

Transitivity of universally-quantified implication

∀x∈D, (P(x)⇒Q(x))∧(Q(x)⇒R(x)) ⇒ ∀x∈D, P(x)⇒R(x)

i.use connectives to show it is a tautology(always true)

ii.use de Morgan, distributive to show the negation of this sentence is a contradiction.

∀x∈D, (P(x) ⇒ (Q(x)⇒R(x))) ⇔ ∀x∈D, ((P(x)∧Q(x)) ⇒ R(x))

use truth table to prove

Multiple Quantifiers

Mixed Quantifiers

If you mix the order of existential and universal quantifiers, you may change the meaning of a sentence.

Operator Precedence: (in decreasing order) 1.¬ 2.∨, ∧ 3.⇒, ⇔ 4.∀, ∃

Proof

Algorithm Analysis and Asymptotic Notation

Computability Theory

Tutorial Exercise 7

For functions f, g: N→R+:

g∈O(f) IFF ∃c∈R+, ∃n_0∈R+, ∀n∈N, n≥n_0 IMPLIES g(n)≤c·f(n)

g∉O(f) IFF NOT(g∈O(f)) IFF ∀c∈R+, ∀n_0∈R+, ∃n∈N, n≥n_0 AND g(n)>c·f(n)

Question 1: Prove that $5n^4 - 3n^2 + 1 \in O(6n^5 - 4n^3 + 2n)$

By the definition of big-Oh, need to show that $\exists c \in R+, \exists n_0 \in R+, \forall n \in N, n \geq n_0$ IMPLIES $5n^4 - 3n^2+1 \leq c \cdot (6n^5-4n^3+2n)$

Discussion:

Working FORWARD from the LHS, $5n^4 - 3n^2+1 \leq 5n^4+1 \leq 5n^4+n^4$ (if $n \geq 1$) $\leq 6n^4 \leq n \cdot n^4$ (if $n \geq 6$) $\leq n^5$

Note that there are other inequality we could have reached, e.g. $6n^4 \leq 6n^5$ for all $n \geq 1$.

Working BACKWARD from the RHS, $6n^5-4n^3+2n \geq 6n^5-4n^3 \geq 6n^5-4n^5$ (because $-n^3 \geq -n^5$) $\geq 2n^5 \geq n^5$

So both chains of inequality connect, we are done and we can pick n0=6 (requires $n \geq 6$ in the first chain) and c=1

Proof: Let c=1, n0=6 and let $n \in N$ be such that $n \geq n_0$:

Then $5n^4 - 3n^2+1 \leq 5n^4+1$
$\leq 5n^4+n^4$ (since $n \geq 6 > 1$)
$\leq 6n^4$
$\leq n \cdot n^4$ (since $n \geq 6$)
$\leq 2n^5$
$\leq 6n^5-4n^5$
$\leq 6n^5-4n^3$ (since $-n^3 \geq -n^5$)
$\leq 6n^5-4n^3+2n$

Question 2: Prove that $6n^5-4n^3+2n \notin O(5n^4-3n^2+1)$

Discussion: The property of n that will be most difficult to show is $6n^5-4n^3+2n > c \cdot (5n^4-3n^2+1)$, so we focus on it first. It is tempting to try to solve for n—and if the expression were simpler, this would yield and appropriate value. But it will be complicated in this case and it is not necessary. Remember that, intuitively, we are simply trying to prove that $6n^5-4n^3+2n$ is larger than $5n^4-3n^2+1$ by more than a constant factor.

Working FORWARD from the LHS, $6n^5-4n^3+2n > 6n^5-4n^3$ (if $n \geq 1$) $\geq 6n^5-4n^5$ (if $n \geq 1$)$=2n^5$

Working BACKWARD from the RHS, $5n^4-3n^2+1 < 5n^4+1$ (if $n \geq 1$) $\leq 6n^4$ (if $n \geq 1$)

Now we want $2n^5 > c(6n^4)$, i.e. $n^5 > 3c(n^4)$.

This will be true as long as $n > 3c$. Since $c \in R+$, to ensure $n \in N$, we can pick $n \geq \text{ceil}(3c)$. This guarantees $n \geq 1$, which is needed for the inequalities above to hold. Finally, we also need $n \geq n0$, which can be achieved simply by picking n=n0+ceil(3c)

Proof: let $c \in R$ and $n0 \in R+$. Let n=ceil(n0)+ceil(3c). Note that $n \in N$ and $n \geq n0$.

Also $6n^5-4n^3+2n > 6n^5-4n^3$ (since $n \geq 1$) $\geq 6n^5-4n^5$ (since $n \geq 1$) $= 2n^5 = n(2n^4)$
$> 3c(2n^4)$ (since $n > 3c$) $= c(6n^4) \geq c(5n^4+1) \geq c(5n^4-3n^2+1)$

Question 3: Prove that $\forall a, b \in R, (1 \leq a \leq b)$ IMPLIES $n^a \in O(n^b)$

Question 4: Prove that $\forall a, b \in R, (1 \leq a \leq b)$ IMPLIES $a^n \in O(b^n)$

Induction

Simple Induction

Principle of Simple Induction: $[P(0)\wedge(\forall n\in N, P(n)\Rightarrow P(n+1))] \Rightarrow \forall n\in N, P(n)$

If the initial case works, and each case that works implies its successor works, then all cases work.

Simple Induction Outline:

state claim P(n)
prove base case first
state inductive hypothesis
prove the inductive step

Inductive Step: state inductive hypothesis H(n)
Derive conclusion C(n): show that C(n) follows from H(n), indicating where you use H(n) and why that is valid
Verify Base Case(s): verify that the claim is true for any cases not covered in the inductive step

Note: in simple induction H(n) is the claim you intend to prove about n, and C(n) is the same claim about n+1 — "simple" because the reasoning moves from n to n+1.

Claim: Every set with n elements has exactly 2^n subsets.

{7, 13, 19} P- {}, {7}, {13}, {7,13} P+ {19}, {7, 19}, {13, 19}, {7,13, 19}
Inductive Step: Let $n\in N$ (let n be an arbitrary natural number)
Assume H(n): Assume every set of size n has 2n subsets.
Show H(n)⇒C(n): every set of size n+1 has 2n+1 subsets.
Let S be an arbitrary set of size n+1.
Let $x\in S$ (we can do this because $n+1\geq 1$ or $S\neq\emptyset$)
Partition the subsets of S into P-(the ones without x), and P+(the ones with x)
Notice that P- is all the subsets of S-{x} (size n), so $|P-|=2n$.
Each subset in P+ pairs with a subset in P-, where they are matched by adding/removing x, $|P+|=|P-|=2n$.
P+ and P- are all subsets of S, so S has 2n+2n=2n+1 subsets.
Base Case: a set of size 0 has 1 subset, $2^0=1$ subset
By Induction, conclude that a set of size n, $n\in N$ has 2n subsets.

Claim: 12^n-1 is a multiple of 11. Prove by induction that $\forall n\in N$ P(n): $12^n-1=11\cdot k$ for some $k\in N$

Base Case: Prove P(0), $12^0-1=1-1=0=11\times 0$ so this holds with k=0
Inductive Hypothesis: Suppose $n\in N$ and P(n), i.e. $\exists k\in N, 12^n-1=11\cdot k$
Inductive Step: Prove there exists k' such that $12^{n+1}-1=11\cdot k'$
$12^{n+1}-1=12\cdot 12^n-1=11\cdot 12^n+12^n-1=11\cdot 12^n+11\cdot k=11(12^n+k)$
conclude that $\exists k'\in N$ such that $12^{n+1}-1=11\cdot k'$ because we can choose $k'=12^n+k$
By Induction, $\forall n\in N$ P(n) i.e 12^n is a multiple of 11 for all n

Claim: $3n\geq n3$ (Pascal's Triangle, Binomial Theorem)

Inductive Step: Assume $n\in N$ and $n\geq 3$,
Assume H(n): $3^n\geq n^3$,
Show that H(n)⇒C(n): $3^{n+1}\geq(n+1)^3$,
$3^{n+1}=3\cdot 3^n\geq 3\cdot n^3$ (by H(n))$=n^3+n^3+n^3\geq n^3+3n^2+9n$ ($n^3\geq 3n^2$, $n^3\geq 9n$, $6n\geq 1$)$=n^3+3n^2+3n+6n\geq n^3+3n^2+3n+1=(n+1)^3$
So C(n) follows from H(n)
Verify Base Case: $3^3\geq 27=3^3$, so the claim holds for 3
Also, $3^0=1\geq 0=0^3$, $3^1=3\geq 1=1^3$, $3^2=9\geq 8=2^3$ (we need this to conclude ∀)
Conclude $3n\geq n3$ ∀n

Exercise 1: $\forall n\in N$, 7^n-1 is a multiple of 6.

Inductive Step: Let n be an arbitrary natural number.
Assume H(n): 7^n-1 is a multiple of 6.
show that C(n) follows from H(n): Denote by C(n) the claim: $7^{n+1}-1$ is a multiple of 6.
Let $k\in Z$, such that $7^n-1=6k$ # since by H(n), 7^n-1 is a multiple of 6
$7^{n+1}-1=7(7^n-1)+6=7(6k)+6=6(7k+1)$ $7k+1\in Z$ # since 7, k, $1\in Z$ and Z closed under +,×
$7^{n+1}-1$ is a multiple of 6, that is C(n) follows from H(n).
Base Case: $7^0-1=1-1=0=6\times 0$, so the main claim holds for natural number 0.

Let P(n) denote $7^n-1=6m_n$, where $m_n \in Z$.
Basis Step: P(0) holds because $7^0-1=1-1=0=6\times0$
Inductive Step: Assume P(k) holds for some arbitrary $k\geq0 \in N$, that is $7^k-1=6m_k$, where $m_k \in Z$
Using the IH, we must show that P(k+1) holds too:

$7^{k+1}-1=7(7^k-1)+6=7(6m_k)+6=6(7m_k+1)$

$m_{k+1}=7m_k+1 \in Z$ since $7, m_k, 1 \in Z$ and Z closed under +,×.

This completes the inductive step.

Hence, $\forall n \in N$, 7^n-1 is a multiple of 6.

Exercise 2: $\forall n \in N$, the units digit of 7^n is in $\{1,3,7,9\}$

Sample solution 1: Proof by simple induction.

inductive step: Let n be an arbitrary natural number. Assume $H(n)$: the units digit of 7^n is in $\{1,3,7,9\}$.

show $C(n)$ follows from $H(n)$: Denote by $C(n)$ the claim: The units digit of $7^{n+1} \in \{1,3,7,9\}$. By $H(n)$ there are natural numbers i,j such that $7^n = 10i+j$ and $j \in \{1,3,7,9\}$. There are four cases to consider:

Case $j=1$: $7^{n+1} = 7\times7^n = 10\times7i+7$, and the units digit $7 \in \{1,3,7,9\}$.

Case $j=3$: $7^{n+1} = 7\times7^n = 70i+21 = 10(7i+2)+1$ and the units digit $1 \in \{1,3,7,9\}$.

Case $j=7$: $7^{n+1} = 7\times7^n = 70i+49 = 10(7i+4)+9$, and the units digit $9 \in \{1,3,7,9\}$.

Case $j=9$: $7^{n+1} = 7\times7^n = 70i+63 = 10(7i+6)+3$, and the units digit $3 \in \{1,3,7,9\}$.

In every possible case, the units digit of $7^{n+1} \in \{1,3,7,9\}$, so $C(n)$ follows from $H(n)$.

base case: $7^0 = 1 \in \{1,3,7,9\}$, so the main claim is verified for natural number 0.

Sample solution 2: Proof by simple induction. We also use the modular arithmetic notation.

Let $P(n)$ denote $7^n \equiv 1,3,7,$ or $9 \pmod{10}$.

basis step: $P(0)$ holds because $7^0 \equiv 1 \pmod{10}$.

inductive step: Assume $P(k)$ holds for some arbitrary $k \geq 0 \in \mathbb{N}$, that is $7^k \equiv 1,3,7,$ or $9 \pmod{10}$.

Using the I.H., we must show that $P(k+1)$ holds too: since $7^{k+1} \equiv 7^k \times 7 \pmod{10}$, there are four cases to consider:

Case $7^k \equiv 1$ (mod 10): By the I.H, $7^{k+1} \equiv 1\times7 \pmod{10} \equiv 7 \pmod{10}$.

Case $7^k \equiv 3$ (mod 10): By the I.H, $7^{k+1} \equiv 3\times7 \pmod{10} \equiv 1 \pmod{10}$.

Case $7^k \equiv 7$ (mod 10): By the I.H, $7^{k+1} \equiv 7\times7 \pmod{10} \equiv 9 \pmod{10}$.

Case $7^k \equiv 9$ (mod 10): By the I.H, $7^{k+1} \equiv 9\times7 \pmod{10} \equiv 3 \pmod{10}$.

This completes the inductive step.

Hence, $\forall n \in \mathbb{N}$, $7^n \equiv 1,3,7$ or $9 \pmod{10}$.

Exercise 3: $\forall n \in N-\{0,1,2,3\}$, $4^n \geq n^4$.

inductive step: Let $n \in \mathbb{N}$. Assume $n \geq 4$ and assume $H(n)$: $4^n \geq n^4$.

show $C(n)$ follows from $H(n)$: Denote by $C(n)$ the claim: $4^{n+1} \geq (n+1)^4$.

$$\begin{aligned} 4^{n+1} &= 4\times4^n \geq 4\times n^4 \# \text{ since by } H(n), 4^n \geq n^4 \\ &= n^4+n^4+n^4+n^4 \\ &\geq n^4+4n^3+16n^2+64n \# \text{ since } n \geq 4 \\ &\geq n^4+4n^3+6n^2+4n+60n \\ &> n^4+4n^3+6n^2+4n+1 \# \text{ since } n \geq 4 > 1/60 \\ &= (n+1)^4 \# \text{ binomial theorem} \end{aligned}$$

That is, $C(n)$ follows from $H(n)$.

base case: $4^4 = 64 \geq 64 = 4^n$, so the main claim holds for natural number 4.

Sample solution 2: Proof by simple induction.

Let $P(n)$ denote $4^n \geq n^4$.

basis step: $P(4)$ holds because $4^4 \geq 4^4$.

inductive step: Assume $P(k)$ holds for some arbitrary $k \geq 4 \in \mathbb{N}$, that is $4^k \geq k^4$.

Using the I.H., we must show that $P(k+1)$ holds too:

$$\begin{aligned} 4^{k+1} &= 4\times4^k \geq 4\times k^4 \# \text{ since by I.H., } 4^k \geq k^4 \\ &= k^4+k^4+k^4+k^4 \\ &\geq k^4+4k^3+16k^2+64k \# \text{ since } k \geq 4 \\ &\geq k^4+4k^3+6k^2+4k+60k \\ &> k^4+4k^3+6k^2+4k+1 \# \text{ since } k \geq 4 > 1/60 \\ &= (k+1)^4 \# \text{ binomial theorem} \end{aligned}$$

This completes the inductive step.

Hence, $\forall n \geq 4 \in \mathbb{N}$, $4^n \geq n^4$.

Complete/Strong Induction

Principle of Complete/Strong Induction: $(\forall n\in N, \{P(0), \dots P(n-1)\}\Rightarrow P(n)) \Rightarrow \forall n\in N, P(n)$

If all previous cases always implies the current case, then all cases are true.

Complete Induction Outline:

Inductive Step: state inductive hypothesis H(n)

Derive conclusion C(n): show that C(n) follows from H(n), indicating where you use H(n) and why that is valid

Verify Base Case(s): verify that the claim is true for any cases not covered in the inductive step

The outline is the same as simple induction, but inductive hypothesis H(n) are modified so that it assumes the main claim for every natural number from the starting point up to n-1, and the conclusion C(n) is now the main claim for n.

Note: in SI, H(n) assumes for one value (for some $n\in N$); in CI, H(n) assumes for $0\le i<n$

Claim: f(n) is a multiple of 3, where f(n)= …

$f(0)=1$ ✘ $f(1)=1$ ✘ $f(2)=(f(1))^2+2\cdot f(1)=3$ ✓ $f(3)=3$ ✓ (because $\lfloor 2\rfloor=\lfloor 3\rfloor$)

$f(4)=(f(2))^2+2\cdot f(2)=15$ ✓ $f(5)=f(6)=f(7)=f(8)=15$ ✓ (because $\lfloor 4\rfloor=\lfloor 5\rfloor=\lfloor 6\rfloor=\lfloor 7\rfloor=\lfloor 8\rfloor$)

$f(9)=(f(3))^2+2\cdot f(3)=15$ ✓ $f(16)=(f(4))^2+2\cdot f(4)=255$ ✓

Inductive Step: Let $n\in N, n\ge 2$

Assume inductive hypothesis H(n): for $i\in N, 2\le i<n$, f(i) is a multiple of 3 (Want to use this)

Show H(n)→C(n): f(n) is a multiple of 3.

Both f(2) and f(3) depend on f(1), which is not assumed in H(n)

Let $n\ge 4$, $f(n)=(f(\lfloor\sqrt{n}\rfloor))^2+2\cdot f(\lfloor\sqrt{n}\rfloor)$ (definition of f(n))

$2\le\sqrt{n}$ (because $n\ge 4$), $\sqrt{n}<n$ (because $n\ge 4>1$), thus $2\le\lfloor\sqrt{n}\rfloor\le\sqrt{n}<n$ (definition of floor). So $f(\lfloor\sqrt{n}\rfloor)$ is a multiple of 3 by H(n).

Let $k\in N$ such that $3\cdot k=f(\lfloor\sqrt{n}\rfloor)$, so $f(n)=(3\cdot k)^2+2\cdot(3\cdot k)=9k^2+6k=3\cdot(3k^2+2k)$

$2, 3, k\in N$ and N is closed under addition & multiplication, so $(3k^2+2k)\in N$ ($N\subset Z$)

f(n) is a multiple of 3 ($n\ge 4$)

Verify Base Cases: 2 and 3 (0,1→Not True, 4→Proven from assuming H(n))

$f(2)=(f(1))^2+2\cdot f(1)=3$ $f(3)=(f(1))^2+2\cdot f(1)=3$ So claim holds for 2 and 3

Note that Base Case is to prove anything not covered in IS, it is not necessary needed.

Conclude: f(n) is a multiple of 3, $\forall n\in N, n\ge 2$.

Claim: Zero Pair Free Binary Strings zpfbs(n) is zp(n)

Denote by zpfbs(n) the number of binary strings of length n that contain no pairs of adjacent zeros. What is zpfbs(n)?

$n=0\rightarrow 1$ empty string

$n=1\rightarrow 2$ "1" "0"

$n=2\rightarrow 3$ "10" "01" "11"

$n=3\rightarrow 5$ "011" "101" "010" "110" "111"

$n=4\rightarrow 8$ "0110" "1010" "1111" "0101" "0111" "1011" "1101" "1110"

constructive idea: n=4 from n=2 "10**10**" "01**10**" "11**10**"

and n=3 "011**1**" "101**1**" "010**1**" "110**1**" "111**1**"

zp(n)= 1 if n=0, 2 if n=1, zp(n-1)+zp(n-2) if $n\ge 2$

Inductive Step: Let $n\in N$

Assume H(n): $\forall i\in N, 0\le i<n$, zp(i) is the number of zpfbs of length i.

Show H(n)→C(n): zp(n) is the number of zpfbs of length n

Let $n\ge 2$, $0\le n-2<n-1<n$ (assumed in H(n)),

Partition the zpfbs of length n into two

P1: those strings that end in 1

P0: those strings that end in 0

$|P1|=zp(n-1)$, by H(n), since $0\le n-1<n$, and there are exactly the zpfbs of length n-1 with "1" appended

$|P0|=zp(n-2)$, by H(n), since $0\le n-2<n$, and there are zpfbs of length n-2 with "10" appended

So the number of zs of length n is $|P1|+|P0|=zp(n-1)+zp(n-2)=zp(n)$ as claimed.

Verify Base Case: zpfbs(0)=1=zp(0) because the empty string has no zeroes.

zpfbs(1)=2=zp(1) because the strings are "1" and "0"

The claim holds $\forall n\in N$

Note: base case is anything not covered in the inductive step, may not have an explicit base case with complete induction.

Claim: every natural number greater than 1 has a prime factorization

integer m is divisible by integer n, n is a divisor of m, if m/n is an integer ($\exists k \in N, m=k*n$)
integer n is prime if $n \geq 2$ and n has no divisor in the set {2, …n-1} (n's only divisor are 1 and n)
a prime factorization of an integer n is a sequence of prime numbers whose product equals n.
84=2*2*3*7 (it is s sequence, it can contain the same number multiple times)
Prove by CI (modified to conclude for $n \geq 2$) that $\forall n \in N, n \geq 2$, n has a prime factorization (PF)
Inductive Step: let $n \in N$ such that $n \geq 2$,

Assume H(n): $\forall i \in N, 2 \leq i < n$, i has a PF
Show H(n)→C(n): n has a PF

Case 1: n is prime, then n is its own PF
Case 2: n is composite, then it has a divisor a such that $a \geq 2$ and $a < n$.

By H(n), a and b have a PF, a=p1…pj (product of primes), b=q1…qk (ps, qs are prime)
Then n=p1…pj·q1…qk, So C(n) follows from H(n)

Conclude by CI, $\forall n \in N, n \geq 2$, n has a PF. (NO explicit base case)

Claim: after a certain natural number n, every postage can be made up by combining 3- and 5- cent stamps

Exercise 1: Full binary trees are binary trees where all internal nodes have 2 children. Prove that any full binary tree with more than 1 node has no more than twice as many leaves as internal nodes. Use complete induction on the total number of nodes.

Proof by Complete Induction on the number of nodes in the binary tree.
Inductive Step: Let $n \in N, n > 1$.

Assume H(n): $\forall i \in N, n > i \geq 2$ every full binary tree with i nodes has no more than twice as many leaves as internal nodes.
Show C(n): Every full binary tree with n nodes has no more than twice as many leaves as internal nodes.
Case: **no full binary trees with n nodes exist**. C(n) is vacuously true (e.g. n=2 or other even numbers greater than 0)
Case: **both of T's subtrees are single-node trees**. T has one interior node (the root) and two leaves, and $2 \leq 2 \times 1$, so C(n) is true
Case: **one of T's subtree**. Let the number of internal nodes in T's subtree with more than 1 node be i_c, and the number of leaves be lc. Since this subtree has fewer than n and more than 1 node, by H(n) we know that $l_c \leq 2i_c$. The number of leaves in T are lc+1, the leaves of the subtree with more than 1 node and the single-node subtree. The number of internal nodes in T are i_c+1, the internal nodes of the subtree with more than one node plus the root. Summing leaves and internal nodes and comparing them we get: $l_c+1 \leq 2i_c+1 \leq 2(i_c+1)$. So C(n) is true in this case.
Case: **both of T's subtrees have more than 1 node**. Let the number of internal nodes and leaves of the left subtree be iL and lL, respectively. Let the number of internal nodes and leaves of the right subtree be i_R and l_R, respectively. Since each subtree has fewer than n and more than 1 node, by H(n) we know that $l_L \leq 2i_L$ and $l_R \leq 2i_R$. Summing T's leaves, and comparing them to number of T 's internal nodes, i_L+i_R+1 (the root is an internal node): $l_L+l_R \leq 2i_L+2i_R = 2(i_L+i_R) \leq 2(i_L+i_R+1)$. So C(n) is true in this case.

In every possible case C(n) follows from H(n).

Exercise 2: Use Complete Induction to show that postage of exactly n cents can be made using only 6-cent and 7-cent stamps, for every natural number n greater than k (you will have to discover the value of k).

Proof by Complete Induction that $\forall n \in N, n > 29$ postage of n cents can be made with 6-cent and 7-cent stamps (guessed k=29 by experimenting)

Inductive Step: Let $n \in N, n > 29$.

Assume H(n): $\forall i \in N, n > i \geq 30$, postage of i cents can be made with 6-cent and 7-cent stamps.
Show C(n): postage of n cents can be made with 6-cent and 7-cent stamps.

Case n = 30 (**base case**): Postage of 30 cents can be formed with 5 6-cent stamps and 0 7-cent stamps.
Case n = 31 (**base case**): Postage of 31 cents can be formed with 4 6-cent stamps and 1 7-cent stamps.
Case n = 32 (**base case**): Postage of 32 cents can be formed with 3 6-cent stamps and 2 7-cent stamps.
Case n = 33 (**base case**): Postage of 33 cents can be formed with 2 6-cent stamps and 3 7-cent stamps.
Case n = 34 (**base case**): Postage of 34 cents can be formed with 1 6-cent stamp and 4 7-cent stamps.
Case n = 35 (**base case**): Postage of 35 cents can be formed with 0 6-cent stamps and 5 7-cent stamps.

Case n > 35: Since n-6 > 29 (subtract 6 from the inequality), we know that 29 < n-6 < n and by H(n) postage of n-6 cents can be formed with 6-cent and 7-cent stamps. Add one 6-cent stamp to make postage for n cents with 6-cent and 7-cent stamps.

In every case C(n) follows from H(n).

Structural Induction

Define sets recursively…
Consider the following set S:
1.$0 \in S$
2.$n \in S \Rightarrow n+1 \in S$
3.S contains nothing else (i.e S is the smallest such set)
This is one way to define the natural number N. That is S=N.
Every element of S is a natural number (by construction, and smallest)
If not smallest —> S=R,Z,Q (supersets of N)
Every natural number is an element of S (every natural number is 1 more than another, except 0)
By smallest, we mean N has no proper subsets that satisfy these conditions. If we leave out smallest, what other sets satisfy the definition?

We defined the simplest natural number (0) and the rule to produce new natural numbers from old (add 1). Proofs using Simple Induction work by showing that 0 has some property, and then that the rule to produce natural numbers preserves the property, that is
1.Show that P(0) is true for basis, 0
2.Prove that $\forall n \in N$, $P(n) \Rightarrow P(n+1)$
The same structure applies if our set is defined differently, e.g. $S=\{n \in N \mid n \geq 2\}$

Structurally-Defined Sets
Define ε: the smallest set such that

x, y, z ∈ε
$e1, e2 \in \varepsilon \Rightarrow (e_1+e_2), (e_1-e_2), (e_1 \times e_2)$ and $(e_1 \div e_2) \in \varepsilon$

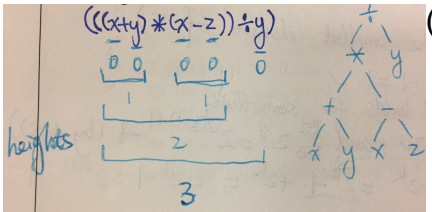
Form some expressions in ε. Count the number of variables (symbols from {x, y, z}) and the number of operators(symbols from {+,-,×,÷}). Make a conjecture about the number of variables and the number of operators in an expression in ε.

Structural Induction Outline:
To prove that a property is true for all e∈ε, parallel the recursive set definition:
Verify Base Case(s): show that the property is true for the simplest members, {x, y, z}, that is show P(x), P(y) and P(z)
Inductive Step: Let e1 and e2 be arbitrary elements of ε.

Assume $H(\{e_1, e_2\})$: P(e1) and P(e2), that is e1 and e2 have the property.
Show that $C(\{e_1, e_2\})$ follows:

All possible combinations of e1 and e2 have the property, that is $P((e_1+e_2))$, $P((e_1-e_2))$, $P((e_1 \times e_2))$ and $P((e_1 \div e_2))$.

Prove ∀e∈ε, P(e) by structural induction (P(e): vars(e)=ops(e)+1)
Verify Base Case: choose e∈{x, y, z}, then vars(e)=1 and ops(e)=0, so vars(e)=ops(e)+1 for e∈{x, y, z}
Inductive Step: Let e1, e2∈ε (2 expressions from set ε)
Assume $H(\{e_1, e_2\})$: vars(e1)=ops(e1)+1 and vars(e2)=ops(e2)+1
Show $H(\{e_1, e_2\}) \rightarrow C(\{e_1, e_2\})$: vars((e1⊙e2)) = ops((e1⊙e2))+1, for all ⊙ in{+,-,×,÷}
vars((e1⊙e2))=vars(e1)+vars(e2) (because no variables are added/removed)
=ops(e1)+1+ops(e2)+1 (by inductive hypothesis H({e1, e2}))
=ops((e1⊙e2))+1 (ops((e1⊙e2))=ops(e1)+ops(e2)+1 because we added the ⊙ operator)
Then our conjuncture holds, vars(e)=ops(e)+1, ∀e∈ε (structurally-defined set ε)
Define the height of x, y, or z as 0, and h((e1⊙e2)) as 1+max(h(e1), h(e2)), if e1, e2∈ε and ⊙∈{+,-,×,÷}.
Prove ∀e∈ε, P(e): $vars(e) \leq 2^{h(e)}$
Let internal nodes be operators and roots be variables:



(((x+y)*(x-z)))÷y
Verify Base Cases: Let e=x, then vars(e)=1 and $2^{h(e)}=2^0=1$, so $vars(e) \leq 2^{h(e)}$
(the same holds for e=y, e=z)

Carrie Yan

Inductive Step: Let $e1, e2 \in \varepsilon$

Assume H($\{e_1, e_2\}$): vars(e1)$\leq 2^{h(e1)}$ and vars(e2)$\leq 2^{h(e2)}$

Show H($\{e_1, e_2\}$)$\rightarrow$C($\{e_1, e_2\}$): vars((e1$\odot$e2))$\leq 2^{h((e1 \odot e2))}$, for all $\odot \in \{+,-,\times,\div\}$

vars((e1$\odot$e2))=vars(e1)+vars(e2) (did not add or remove any variables)

$\leq 2^{h(e1)}+2^{h(e2)}$ (by H($\{e_1, e_2\}$)) $\leq 2^{\max(h(e1),h(e2))}+2^{\max(h(e1),h(e2))} = 2^{\max(h(e1),h(e2))+1} = 2^{h((e1 \odot e2))}$ (by definition of height)

Conclude, by structural induction, that vars(e)$\leq 2^{h(e)}, \forall e \in \varepsilon$

Exercise 1: define the set of expressions E as the smallest set such that:

(a)x, y, z$\in$E (b)If $e_1, e_2 \in$ E, then so are (e_1+e_2) and $(e_1 \times e_2)$.

Define p(e) : Number of parentheses in e

Define s(e) : Number of symbols from {x, y, z, +, ×} in e, counting duplicates

Use structural induction to prove that for all e$\in$E, p(e) = s(e)-1.

Predicate P(e): p(e)=s(e)-1

Verify Basis: the basis elements are three symbols x, y, z. For e$\in${x, y, z} the solitary symbol means s(e)=1. There are no parentheses, so p(e)=0, and p(e)=0=1-1=s(e)-1. So for any e in the basis, P(e)

Inductive Step: Let e_1, e_2 be arbitrary elements of E. Assume H($\{e_1, e_2\}$): P(e_1) and P(e_2), that is p(e_1)=s(e_1)-1 and p(e_2)=s(e_2)-1.

Derive C($\{e_1, e_2\}$): P((e_1+e_2)) and P(($e_1 \times e_2$)), to show p((e_1+e_2))=s((e_1+e_2))-1 and p(($e_1 \times e_2$))=s(($e_1 \times e_2$))-1

Let $\odot \in \{+, \times\}$,

p(($e_1 \odot e_2$))=p(e_1)+p(e_2)+2 since the new expression adds 2 parentheses to those contained in e1 or e2.

s(($e_1 \odot e_2$))=s(e_1)+s(e_2)+1, since the new expression has all the symbols of e1 and e2, plus $\odot$.

Put these ideas together: p(($e_1 \odot e_2$))=p(e_1)+p(e_2)+2=s(e_1)-1+s(e_2)-1+2 (by H($\{e_1, e_2\}$))=s(e_1)+s(e_2) (regrouping)=s(($e_1 \odot e_2$))-1

So P(($e_1 \odot e_2$)). Since $\odot$ is an arbitrary element of {+, ×}, this establishes C($\{e_1, e_2\}$).

Exercise 2: define the set of non-empty full binary trees T, as the smallest set such that:

(a)Any single node is an element of T.

(b)If $t_1, t_2 \in$ T, then so is any root node with edges to t_1 and t_2.

Use structural induction to prove that any non-empty full binary tree has an odd number of nodes.

Predicate P(t): t has an odd number of nodes

Basis: the basis consists of single-node FBTs, hence every element of the basis has an odd number of nodes (i.e. 1)

Inductive Step: Let t_1, t_2 be arbitrary elements of T.

Assume H($\{t_1, t_2\}$): P(t_1) and P(t_2), that is t_1 and t_2 each have an odd number of nodes.

Derive C($\{t_1, t_2\}$): Let t be a tree formed by an arbitrary root edges to t1 and t2.

Then P(t) (t has an odd number of nodes.)

Let $k_1, k_2 \in$ N such that t1 has $2k_1+1$ nodes and t_2 has $2k_2+1$ nodes.(By P(t_1) and P(t_2), each tree has an odd number of nodes)

The number of nodes in t is the sum of the nodes in t_1 and t_2: $1+2k_1+1+2k_2+1=2(k_1+k_2+1)+1$.

This is an odd number since $(k_1+k_2+1) \in$ N, due to $k_1, k_2, 1, 2 \in$ N and N being closed under +, ×.

So P(t), which establishes C($\{t_1, t_2\}$).

Well-Ordering

Principle of Well-Ordering: Every non-empty subset of N contains a smallest element.

—Applies to both finite and infinite subsets of N

—A property of N (does not apply to sets like Z, Q, R)

subset $2 \leq i \leq 6$, $i \in$ Z ✓ $i \in$ Z, i is odd ✘ $2 < i < 6$, $r \in$ R ✘ (no smallest)

Well-Ordering Proof Outline: template for a proof of $\forall n \in$ N, P(n) using W-O

For a contradiction (or use contradiction later), suppose $\exists n \in$ N, !P(n).

Then, set S={$n \in$ N: !P(n)} is not empty.

By Well-Ordering, S contains a smallest element k.

At this point, we know:

—!P(k) (because k$\in$S)

—P(i), $\forall i \in \{0, \ldots, k-1\}$ (because k is smallest in S)

Use both facts to derive a contradiction.

Hence, by contradiction, $\forall n \in$ N, P(n).

It turns out that simple induction, complete induction, well ordering are all equivalent to each other. You can use any one of them to conclude the other two.

Prove that $\forall n\in N$, $\sum_{i=0}^{n} 2^i = 2^{n+1}-1$

Proof by contradiction: Suppose $\exists n\in N$, $\sum_{i=0}^{n} 2^i \neq 2^{n+1}-1$

Consider $S=\{n\in N: \sum_{i=0}^{n} 2^i \neq 2^{n+1}-1\}$

By our assumption, S is non-empty.
S is a non-empty of N, so by well-ordering, S has a smallest element.
Let k be the smallest element in S.

At this point, we know (i) $\sum_{i=0}^{k} 2^i \neq 2^{k+1}-1$ (because $k\in S$)

(ii) $\sum_{i=0}^{j} 2^i = 2^{j+1}-1$, $\forall j=\{0,\ldots,k-1\}$ (because k is the smallest element of S)

$\sum_{i=0}^{0} 2^i = 2^0 = 1 = 2^{0+1}-1 = 1$ ✓ $k\neq 0$, so (ii) holds for something. So $k>0$, then $k-1\geq 0$, and $\sum_{i=0}^{k-1} 2^i = 2^{(k-1)+1}-1$ (by (ii))

and $\sum_{i=0}^{k} 2^i = \sum_{i=0}^{k-1} 2^i + 2^k = 2^k - 1 + 2^k = 2\cdot 2^k - 1 = 2^{k+1}-1$ which contradicts (i)

There is no $n\in N$ for which $\sum_{i=0}^{n} 2^i \neq 2^{n+1}-1$

Conclude, …

Prove $\forall m, n\in N, n>0, \exists r, \exists q$ (r, $q\in N$), such that $r<n$ and $m=q\cdot n+r$.

Prove that for every pair of natural numbers, m, n, $n>0$. We can do integer division with a remainder $<n$.

Suppose $m, n\in N, n>0$

Consider $R=\{r\in N: \exists q\in N, m=q\cdot n+r\}$, R is not empty, because if $q=0$, then $m\subseteq R$. i.e $m=0\cdot n+m$
So by well-ordering, R has a smallest element.
Let r' be the smallest element of R,

We know: (i)$\exists q$, $m=q\cdot n+r'$
(ii)$\forall s\in\{0, \ldots, r'-1\}$, $\forall q\in N$, $m\neq q\cdot n+s$

Claim: $r'<n$

Proof by contradiction: assume $r'\geq n$, let $q\in N$ such that $m=q\cdot n+r'$ (we know such a q exists by (i))
$m=q\cdot n+r'=q\cdot n+n+r'-n=(q+1)n+(r'-n)$
so $r'-n\in R$ and $r'-n<r'$ (because $n>0$) which contradicts (ii)

Therefore, $\forall m\in N$, $\forall n\in N$, $n>0$, $\exists r\in N$, $\exists q\in N$, $r<n$ and $m=q\cdot n+r$

Analysis of Recursive Algorithm

Use Induction here
—Complexity of Recursive Algorithm
—Correctness of Recursive Algorithm

Recursively Define Function

define $f(n)=\begin{cases} 2 & n=0 \\ 7 & n=1 \\ 2f(n-2)+f(n-1) & n>1 \end{cases}$

n	0	1	2	3	4	5	6	…
f(n)	2	7	11	25	47	97	191	…

Conjecture: $f(n)<2^{n+2}$, $\forall n\in N$

We will use complete induction because f(n) depends on more than f(n-1)

Inductive Step: Let $n\in N$

Assume H(n): $\forall i\in N, 0\leq i<n, f(i)<2^{i+2}$
Show H(n)→C(n): $f(n)<2^{n+2}$
Let $n>1$, By definition, $f(n)=2\cdot f(n-2)+f(n-1) < 2\cdot 2^{(n-2)+2}+2^{(n-1)+2}$ (by H(n), $n>1$, $0\leq n-1$, $n-2<n$)
$=2^{n+1}+2^{n+1}=2^{n+2}$ So C(n) holds.

Base Case: $n=0$, $n=1$, $2<2^{0+2}$, $7<2^{1+2}$

So the claim holds $\forall n\in N$

Algorithm Complexity:
—Running Time T(n) is measured by counting steps in an algorithm
—Sufficient to count chucks of instructions (sequences that are always executed together in constant time) as one step
—Measure as a function T(n) of input size n (number of input elements)
—For now, just concerned with worst-case (maximum over all inputs of the same size)
—Often, there is no simple algebraic expression for T(n) asymptotic notation → bounds on T(n)

Upper Bound: $T(n)\in O(f(n))$ if some $c\in R+$, $B\in N$ such that $T(n)\le c\cdot f(n)$, $\forall n\in N$, $n\ge B$
Lower Bound: $T(n)\in \Omega(f(n))$ if $\exists c\in R+$, $B\in N$ such that $T(n)\ge c\cdot f(n)$, $\forall n\in N$, $n\ge B$
Tight Bound: $T(n)\in \Theta(f(n))$ $T(n)\in O(f(n))$ and $T(n)\in \Omega(f(n))$

Recursive Binary Search: T(n) n=e-b+1

```
RecBinSearch(x, A, b, e):
    if b == e:
        if x<= A[b]:  return b
        else:    return e+1
    else:
        m = (b+e)//2       # midpoint
        if x<=A[m]:
            return RecBinSearch(x, A, b, m)      # recursive calls T(ceil(n/2))
        else:
            return RecBinSearch(x, A, m+1, e)    # recursive calls T(floor(n/2))
```

We should prove m-b+1=ceil(n/2) e-(m+1)+1=floor(n/2)

We can represent T(n) of RecBinSearch as the recurrence $T(n)=\begin{cases} 1 & (really\ \Theta(1)) \\ 1+\max(T(\lceil n/2\rceil)+T(\lfloor n/2\rfloor)) \end{cases}$

How do we solve T(n) to get a closed form representation to move towards finding a bound?
Use **Repeated Substitution/Unwinding** to get a guess.

Recursive Factorial

```
Factorial(n):
    if n==0 or n==1:   # constant time
        return 1
    else:
        return n*Factorial(n-1)
```

Worst-Case Runtime: $T(n)=\begin{cases} 1\ (\Theta(1)) & if\ n=0\ or\ n=1 \\ 1+T(n-1) & if\ n>1 \end{cases}$ closed form?

Solving Recurrence Relations (unwinding T(n)): T(n)=1+T(n-1)=1+1+T(n-2)=1+1+1+T(n-3)
Pattern: after i substitutions, T(n)=i+T(n-i) T(n)=(n-1)+T(n-(n-1))=n-1+1=n So our guess is T(n)=n
Inductive Step: Let $n\in N$, $n\ge 1$
Assume H(n): $T(k)=k$, $1\le k<n$, $k\in N$
Show H(n)→C(n): T(n)=n
Let $n>1$, T(n)=1+T(n-1) (by definition) =1+(n-1) (by H(n), because $n>1$, $1\le n-1<n$) = n
Note: always check the conditions, make sure we can use IH
Conclude C(n).
Base Case: T(1)=1
Conclude T(n)=n, $\forall n\in N$, $n\ge 1$

Recursive Binary Search
Make some simplifying assumptions:

Suppose $n=2^k$, $k\in N$, Then $T(n)=\begin{cases} 1 & n=1 \\ 1+T(n/2) & n>1 \end{cases}$ (because we assumed n is even, when n even, $\lceil n/2\rceil = \lfloor n/2\rfloor$)

$T(n)=T(2^k)=1+T(2^{k-1})=1+1+T(2^{k-2})=1+\ldots+1+T(2^{k-k})$ (want to reach base case)$=k+1=\log_2 n+1$
T(n) cannot be logn+1 for all n, because $T(n)\in N$, and we made simplifications.
Conjecture: $T(n)\in\Theta(\log n)$
Lower Bound: to prove $T(n)\in\Omega(\log n)$, need $\exists c\in R+,\exists B\in N$ such that $T(n)\ge c\cdot\log(n)$, $n\ge B$.

Notes

from pseudocode

$m - b + 1 = \lfloor \frac{e+b}{2} \rfloor - b + 1$ $(\lfloor x \rfloor + k = \lfloor x + k \rfloor$ if $k \in \mathbb{Z})$

$= \lfloor \frac{e+b-2b+2}{2} \rfloor = \lfloor \frac{e-b+1+1}{2} \rfloor = \lceil \frac{e-b+1}{2} \rceil$ $(\lfloor \frac{k+1}{2} \rfloor = \lceil \frac{k}{2} \rceil \ \forall k \in \mathbb{N})$

$= \lceil n/2 \rceil$

$e - m = e - \lfloor \frac{e+b}{2} \rfloor = e + \lceil -\frac{e+b}{2} \rceil$ $(-\lfloor x \rfloor = \lceil -x \rceil)$

$= \lceil e - \frac{e+b}{2} \rceil$ $(\lceil k + x \rceil = \lceil x \rceil + k, k \in \mathbb{Z})$

$= \lceil \frac{e-b}{2} \rceil = \lfloor \frac{e-b+1}{2} \rfloor$ $(\lfloor \frac{k+1}{2} \rfloor = \lceil \frac{k}{2} \rceil, k \in \mathbb{Z})$

$= \lfloor n/2 \rfloor$ (n is the # of elements to search ie. $e - b + 1$)

Pick B=2, c=1, prove by Complete Induction.

Inductive Step: Let $n \in N, n \geq B$

Assume H(n): $T(i) \geq c \cdot \log(i), \forall i, B \leq i < n$

Show $H(n) \rightarrow C(n)$: $T(n) \geq c \cdot \log(n)$

$T(n) = 1 + T(\lceil n/2 \rceil)$ (because T is monotonically increasing)

$\geq 1 + c \cdot \log(\lceil n/2 \rceil)$ (because $n > B > 1$, so $n \geq 2$, $B \leq \lceil n/2 \rceil < n$ by H(n))

$\geq 1 + c \cdot \log(n.2)$ (because log is monotonic increasing) $= 1 + c(\log(n) - \log(2)) = 1 + c \cdot \log(n) - c \cdot 1$ (log identity) $= c \cdot \log(n)$

So C(n) holds.

Base Case: …

Upper Bound: to prove $T(n) \in O(\log n)$, need $\exists c \in R+, \exists B \in N, \forall n \in N$ such that $T(n) \leq c \cdot \log(n), n \geq B$.

B=? c=?

$T(n) = 1 + T(\lceil n/2 \rceil)$

$\leq 1 + c \cdot \log(\lceil n/2 \rceil)$ (want to sat that by H(n), what is $\lceil n/2 \rceil$ less than?)

$\leq 1 + c \cdot \log((n+1)/2) = 1 + c \cdot (\log(n+1) - \log(2)) = 1 + c \cdot \log(n+1) - c$ (problem, we want $c \cdot \log(n)$, need to get rid of +1)

$\lceil n/2 \rceil \leq (n+1)/2$ so $\lceil n/2 \rceil - 1 \leq (n+1)/2 - 1 = (n-1)/2$

Change our proof to show $T(n) \leq c \cdot \log(n-1)$ $[\leq c \cdot \log(n)]$ +2

$T(n) = 1 + T(\lceil n/2 \rceil) \leq 1 + c \cdot \log(\lceil n/2 \rceil - 1) + 2 \leq 1 + c \cdot \log((n-1)/2) + 2 \leq 1 + c \cdot (\log(n-1) - \log(2)) + 2 = 1 + c \cdot \log(n-1) - c + 2$ pick c=1

$= c \cdot \log(n-1) + 2$

Base Case: $T(1) = 1 \leq \log(1-1) = \log(0)$ not defined

So B>1, $T(2) = 1 + T(\lceil 2/2 \rceil) = 1 + T(1) = 2$ while $\log(2-1) = \log(1) = 0$ $2 \geq 0$

What if we add 2? $T(n) \leq \log(n-1) + 2$ in H(n), $\forall i, B \leq i < n, n > 2, \lceil n/2 \rceil \geq B$